Vol.11 No.4 August/September 1992

BEEBUG

FOR THE
BBC MICRO &
MASTER SERIES

Emblems Card Game

- DYNAMIC MEMORY WINDOW
- COLOURFLASH
- DESIGNER CHARACTERS
- FREE-TEXT DATAFILES

BEEBUG Vol.11 No.4 August/September 1992

FEATURES

Designer Characters (1) 7

Emblems 12

Free-Text Datafiles 17

Mr Toad's Macro ROM 19

Wordwise User's Notebook: The Curious Case of the Forgotten Keys 25

First Course: Graphics 29

512 Forum 34

High Security Ciphers (2) 38

Dynamic Memory Window 43

Mr Toad's Machine Code Corner (5) 46

Workshop: Linked Lists (2) 49

ColourFlash 53

Public Domain Software (7) 55

REGULAR ITEMS

Editor's Jottings 4

News 5

RISC User 42

Hints and Tips 59

Personal Ads 60

Postbag 61

Subscriptions & Back Issues 62

Magazine Disc 63

HINTS & TIPS

Fame and Fortune on Film

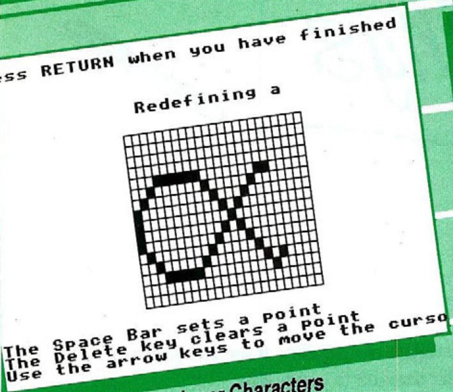
DFS/ADFS Disc Problems

Wordwise Codes

PROGRAM INFORMATION

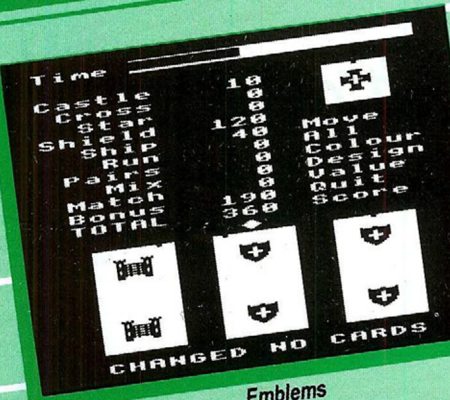
All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



```
ss RETURN when you have finished

Redefining a

The Space Bar sets a point
The Delete key clears a point
Use the arrow keys to move the curso
```

Designer Characters

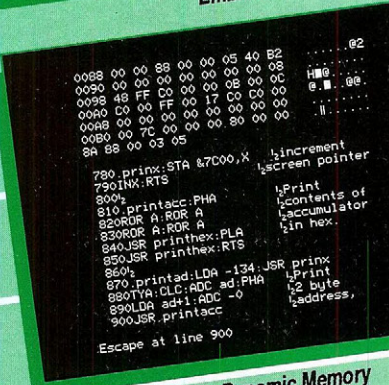


```
Time        10
Castle       0
Cross        0
Star       120   Move
Shield      40   All
Ship         0   Colour
Run          0   Design
Pairs        0   Value
Mix          0   Quit
Match      190   Score
Bonus      360
TOTAL

CHANGED NO CARDS
```

Emblems

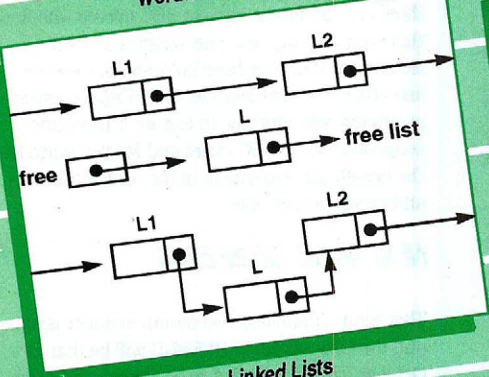
```
▮Start
SHOPPING LIST
-------------
BISON LOTION
SHAMPOO
BEADLE BEE-GON  SPRAY
MR▮TOAD'S WART REMOVER
CAT FOOD
CAT
BEETROOT
▮BEETROOT STAIN REMOVER
OLDE FASHIONED MICROWAVE WAFFLE MIX
WIZ-BANG FLAKES
LOW FAT LARD
BREAD
DRIPPING
RUBBER WASHER
▮End
```

Wordwise Notebook

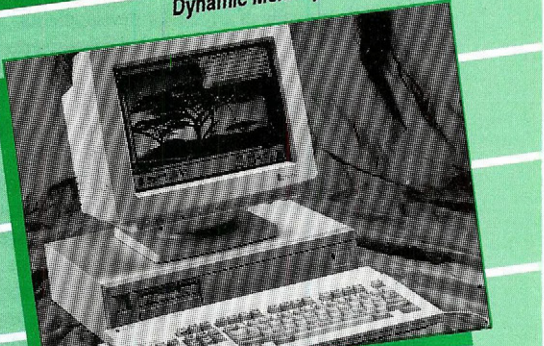


```
0088 00 00 88 00 00 05 40 B2   .......@2
0090 00 00 00 00 00 00 00 08   ........
0098 48 FF C0 00 00 0B 00 0C   H█@.....
00A0 C0 00 FF 00 17 C0 C0 00   @.█..@@.
00A8 00 00 00 00 00 00 00 00   ........
00B0 00 7C 00 00 00 80 00 00   .‖......
8A 88 00 03 05

 780.prinx:STA &7C00,X   ½increment
 790INX:RTS              ½screen pointer
 800½                     ½Print
 810.printacc:PHA         ½contents of
 820ROR A:ROR A           ½accumulator
 830ROR A:ROR A           ½in hex.
 840JSR printhex:PLA
 850JSR printhex:RTS
 860½
 870.printad:LDA -134:JSR prinx
 880TYA:CLC:ADC ad:PHA    ½Print
 890LDA ad+1:ADC -0       ½2 byte
 900JSR printacc          ½address,

Escape at line 900
```

Dynamic Memory



Linked Lists



Upgrading to an Arc

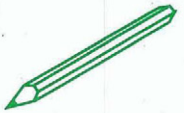
available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings



UPGRADING TO AN ARCHIMEDES

I know that some BEEBUG readers seem to find any mention of the Archimedes range in BEEBUG not to their taste, but it is a fact of life that many BBC micro owners (and BEEBUG readers) have already upgraded to an Archimedes, and no doubt many more will choose to follow this path in the future. Thus the Archimedes range is a legitimate subject for BEEBUG, albeit one which warrants only limited and occasional treatment in these pages.

BEEBUG has a role to play in keeping readers informed generally of any major developments from Acorn, and providing detailed assessments of significant new products such as the A4 notebook computer reviewed last month. For those who may want to consider a move from the 8-bit BBC micro to a 32-bit Acorn RISC the indications are that this autumn could be a good time to make a change, with the expected advent of RISC OS 3 for all of Acorn's Archimedes range, and a wider choice of Archimedes systems than ever before. The Archimedes in its various forms has also been around long enough that there are good supplies of secondhand machines available too.

The back cover of this and future issues of BEEBUG (for the time being) will contain information and advice on upgrading to an Archimedes through Beebug Ltd., our associated company which is one of the major Acorn dealers in the country. Beebug can provide both new and secondhand systems, and can take your existing BBC micro in part exchange. We will also consider more extensive articles in BEEBUG on the subject of upgrading if there is sufficient reader demand to justify this.

However, rest assured that BEEBUG is not about to become an Archimedes-based magazine - there would be no point in this anyway as we already publish a second magazine, RISC User, which is devoted to the Archimedes. All that we intend to do is what we have done since the Archimedes was first launched, and that is to ensure that BEEBUG readers are kept informed of major developments by Acorn, and have the help and advice which they have a right to expect from a magazine such as BEEBUG when it comes to making that heart-in-mouth decision to upgrade.

As has been said on several occasions in the past, there are still many users of BBC micros who find that their existing machine satisfies all their needs, and that they have no desire to consider any other. The vast majority of BEEBUG's pages, as always, will continue to feature articles and programs for the BBC micro and Master range for the benefit and enjoyment of the users of these still excellent machines.

NEXT ISSUE OF BEEBUG

This is our customary two-month summer issue, and the next edition of BEEBUG will be that for October. This is expected to be mailed out at the end of September, in time for the 1992 BBC Acorn User Show. This takes place on the 16th to 18th October at the Wembley Conference Centre. RISC Developments and BEEBUG will be there; we hope you will be too.

M.W.

BEEBUG News

ACORN A4 PC COMPATIBLE

Acorn has confirmed that the top model of the A4 laptop computer, reviewed last issue, will be supplied complete with the latest version of the PC Emulator included in the price of £1699 ex. VAT. Talking of new machines, rumour has it that Acorn will soon release budget priced ARM 3 versions of the A3000 - good news for all budding Archimedes owners if it proves to be true. The A4 laptop, and any other future models in the Archimedes range, will be available through Acorn dealers including Beebug Ltd.

HISTORY SOFTWARE FROM SCET

The Scottish Council for Educational Technology has released another educational software pack, this time for history. The *Bowbridge Pack* is an historical investigation using a database and supporting curriculum materials focussing on the Accidents Register kept at Bowbridge Jute Works in Dundee in 1896 and 1897. This resource allows pupils to access authentic, historical information while developing investigation skills in an historical context.

The Bowbridge Pack is available for the BBC Master (using Key) and for the Archimedes (using Key+), and costs £12.50 ex. VAT from SCET, Dowanhill, 74 Victoria Crescent Road, Glasgow G12 9JN, tel. 041-334 9314, fax 041-334 6519. Scottish schools may obtain the pack through their regional distribution centre.

THE BEEB IS A HIT ON CD

A CD called *Essential HiTech Sound Effects* includes examples of sound samples from the BBC micro. For example, track 10 contains BBC B switch-on beep, disc drive noises, printer sounds, and noises from various Beeb games. The CD costs £11.25 from branches of W.H.Smiths and other suppliers.

PEEK-A-BOO

Sherston Software is another company which is still developing and releasing new educational software for the BBC micro. Latest offerings include *The Three Little Pigs at Home* at £25.00 ex. VAT, *Peek-a-boo around Our House* at £21.50 ex. VAT, and *Peek-a-boo at Opposites* also for £21.50 ex. VAT. All these titles are designed for use with infants, allowing children to explore a story-line and participate in a variety of related activities. For further information contact Sherston Software, Swan Barton, Sherston, Malmesbury, Wilts SN16 0LH, tel. (0666) 840433, fax (0666) 840048.

SNUG IN NEED

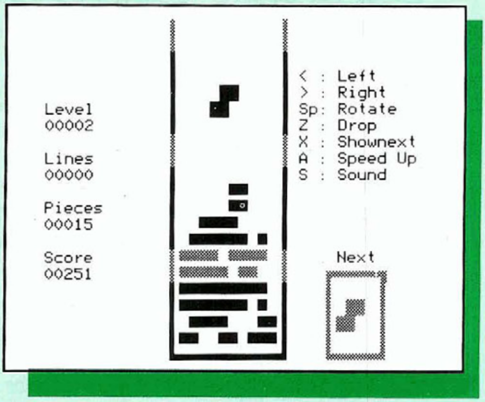
The *Special Needs User Group* is dedicated to the positive uses of computers by and for those people who have special needs. It is organised around a network of local contacts and groups, with additional groups catering for specific interests (communications, dyslexia, physical handicaps, etc.). SNUG (as it is known) also publishes a regular newsletter and journal. The Secretary of SNUG is Jeff Hughes, 39 Eccleston gardens, St Helens WA10 3BJ, tel. (0744) 24608.

ALL FORMATS COMPUTER FAIRS

After a break in August, the All Formats Computer Fairs are back in September and throughout the autumn period. September and early October events are as follows:

5 Sept	National Motorcycle Museum, Birmingham (J6 - M42).
12 Sept	Sandown Racecourse, Esher, Surrey (J9/10 - M25).
19 Sept	Donnington Racecourse (J24 - M1).
3 Oct	Northumbria Centre, Washington, Co. Durham (A194M).
4 Oct	University Sports Centre, Leeds (end M1).

To celebrate the tenth anniversary of the founding of BEEBUG magazine, we have put together a selection of the best programs which we have published over the past ten years. This disc is packed with applications, utilities and games most of which have not been available previously other than when first published in BEEBUG. All the programs come with full on-screen help files, which can be printed out as well for permanent reference.



BEEBUG 10th Anniversary Disc

Celebrate BEEBUG's 10th anniversary with this disc at the special low price of £4.95 and you will have something to celebrate too.

DRAUGHTS - An implementation of the classic board game in which you pit your wits against a computerised opponent.

KEYSTRIP DESIGNER - A very well written program to enable the creation, editing and printing of function key strips.

GARP - GARP (Geographical Atlas using Radial Projection) allows views of the globe to be displayed from any point above the Earth's surface.

MULTI-COLUMN PRINTING - This utility formats any text file into columns, and prints the result using an Epson FX-80 or compatible printer.

PERPETUAL CALENDAR - This program can display or print the calendar month by month for any year between 1753 and 5000 A.D. in the United Kingdom, or even earlier in other countries.

QUAD - Quad is a Tetris-like game, in which you must manipulate falling blocks to slot into each other. Dangerously addictive!

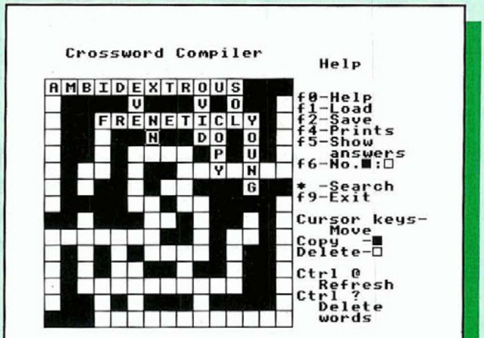
STEALTH - In this game you play against an opponent (or the computer), who sets a number of targets for you to find, and you must use your skill to discover the locations of the targets in as few goes as possible.

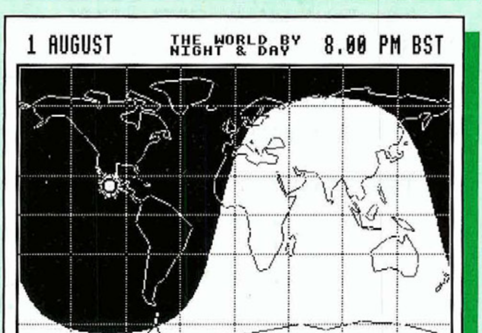
RECURSIVE TREES - This fascinating program uses recursion to create an infinite variety of tree-like designs - you choose a set of numbers, and the computer does the rest.

THE WORLD BY DAY AND NIGHT - This program will draw a map of the world showing graphically where the sun is in the sky or where it's night at every spot on earth.

CROSSWORD COMPILER - This program allows crosswords to be designed and the clues compiled.

PBB3a	*3.5" ADFS*	*£4.95 inc. VAT + £1 p&p*
PBB5a	*5.25" DFS 40/80T*	*£4.95 inc. VAT + £1 p&p*





RISC Developments Limited 117 Hatfield Road, St Albans, Herts AL1 4JS. Tel. (0727) 40303 Fax. (0727) 860263

Designer Characters (Part 1)

Elaine Kemp explains how to modify your printer's fonts.

INTRODUCTION

Some modern dot matrix printers, when fitted with an internal buffer, enable the complete redefinition of both NLQ and draft characters. It is, however, tedious to work out the data needed by the printer for any new characters. The program *Design*, listed at the end of this article, simplifies the design and definition of both types of characters. It could, in fact, be used to redesign an entire font, although its main use will probably be in designing mathematical, scientific or foreign language characters.

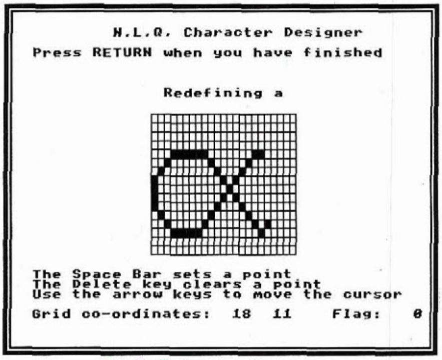
It is important to note that not all printers have the facility to redefine characters so you must consult your manual regarding your own printer. You may also need to check whether any necessary buffer has been fitted, or if a dip switch needs altering.

The program was originally written with a Panasonic KX-P1180 fitted with a 32K RAM chip and may need modification to work on other printers. It should work on all Acorn machines but, because memory is restricted on a standard model B, PAGE must be reset to &1100 before running the program in this case.

ENTERING THE PROGRAM

Because of the large amount of memory used by the program, you should omit all REM statements when typing in the program, (though Master users may include them if they wish). Save the program as *Design*. It is important to check that the codes in *PROCprinter* and *PROCprintgrid* are correct for your own printer, the REM statements will guide you here.

Most printers use a grid of 11 x 8 bits for draft and 23 x 16 for NLQ characters, requiring 12 and 47 data bytes per character respectively. If your printer uses a different matrix, then the variables in line 1050, the number of DATA items in line 2520 and the DATA in lines 2760 and 2770 must be altered.



```
          N.L.Q. Character Designer
  Press RETURN when you have finished

               Redefining a

  The Space Bar sets a point
  The Delete key clears a point
  Use the arrow keys to move the cursor
  Grid co-ordinates:  18  11    Flag:   0
```

Re-defining a character

USING THE PROGRAM

When run, you initially have the option of designing either a draft or NLQ character. You are then presented with the appropriate grid and asked which character you wish to redefine. At this stage you are limited to characters within the ASCII range 32 to 126.

The cursor is a rectangular box which is moved over the grid with the cursor keys. The co-ordinates of the cursor are shown below the grid. The Space Bar sets the point and fills the box in. Similarly, the Delete key unsets the point and clears the box. If an element is set, then its flag is 1; this is shown at the bottom right of the screen. Pressing 'Q' at this point will offer the option of re-running the program or quitting.

You must ensure that you design characters in line with your printer manual's instructions; for instance, some printers do not allow downloaded characters to have two horizontally adjacent dots. Check to see what restrictions apply for your printer. When you are satisfied with your design, press Return.

The program calculates the data bytes for the character and the necessary attribute byte, your manual should explain further. You are asked if you wish to use the upper or lower pins. In general, characters without descenders use the upper pins. You are then asked if you wish to use the full width and you should normally type 'N' unless you are designing a graphics character and do not wish its width to vary in proportional spacing mode. A minimum width of 5 dots is assumed in line 1920. The data for the character is then displayed on the screen and you are asked if you want a hard copy.

If you answer 'Y' the data is printed, then the character downloaded and printed out in both normal and double size. You may optionally print out the grid as well; this is a rapid printout rather than a screen dump. Finally you are offered the option of editing the same character further, redefining another character or quitting the program.

PROGRAM NOTES

It is assumed that for a draft character the data bytes are sent from left to right and that the bottom dot of each column has the lowest bit value. For NLQ each column is divided into two bytes. The first data byte to be sent represents the top eight dots of the first column, the second byte represents the lower eight dots of the first column. *PROCcalc1* calculates the data for draft characters to be sent to the printer and *PROCcalc2* the data for NLQ characters. *PROCbytes* calculates the attribute byte after determining the start and end positions of the character, represented by *F%* and *W%* respectively. Your printer manual should elucidate further. The printout of the grid is accomplished by redefining 0 and 1, selecting the download font, and then printing the flag for each element of the matrix. It was found necessary to include a time delay in lines 2410 and 2610 before switching off the printer with VDU 3, or else not all the data in the buffer reached the printer.

MODIFYING THE PROGRAM

Assuming that your printer will accept downloaded characters, it may still be necessary to modify the program to suit it; look especially at the Escape codes in lines 2240 to 2400. The program is well commented so that you can determine what each code sequence should do. *Q%* represents the number of columns and R% the number of rows in the matrix. The relationships between the variables in line 1050 are given by the formulae:

```
L% = H% + Q%*D%
M% = K% + R%*E%
```

It is not really feasible to alter the value of *R%* to anything other than 8 or 16.

In the next two articles a program will be presented which will give some definitions of some commonly used scientific and mathematical characters, as well as guidelines on designing characters and incorporating them into text files.

```
   10 REM Program Design
   20 REM Version B 1.0
   30 REM Author  Elaine Kemp
   40 REM BEEBUG  Aug/Sep 1992
   50 REM Program subject to copyright
   60 :
  100 ONERROR OSCLI"FX4":OSCLI"FX12":VDU
7,22,7:REPORT:PRINT" at line ";ERL:END
  110 IFPAGE>&1100:VDU7,22,7:PRINT"PAGE
must be set to &1100":END
  120 MODE1:PROCinit:PROCgrid
  130 REPEAT
  140 PROCinfo:PROCdesign
  150 PROCprint:PROCchoose
  160 UNTILV%=TRUE
  170 IFG%=50 THEN RUN
  180 @%=&90A:MODE7:OSCLI"FX12":OSCLI"FX
4":PRINT"Finish"
  190 END
  200 :
 1000 DEF PROCinit
 1010 VDU23,1,0;0;0;0;
 1020 VDU19,1,3,0,0,0,19,2,4,0,0,0,19,3,
6,0,0,0
 1030 COLOUR1:VDU7,31,15,12:PRINT"D";:CO
LOUR2:PRINT"raft?":VDU31,15,14:PRINT"or"
:COLOUR1:VDU31,15,16:PRINT"N";:COLOUR2:P
RINT".L.Q?":*FX15
 1040 REPEAT:G%=INSTR("DdNn",GET$):UNTIL
G%
 1050 IFG%<=2 H%=400:K%=300:L%=840:M%=74
8:Q%=11:R%=8:D%=40:E%=56:N%=0:A$="Draft
" ELSE H%=380:K%=300:L%=840:M%=748:Q%=23
:R%=16:D%=20:E%=28:N%=1:A$="N.L.Q. "
 1060 DIMgrid%(Q%,R%),bytes%(Q%*(N%+1))
 1070 @%=&04:X%=1:Y%=1:V%=FALSE
 1080 CLS:VDU24,H%;K%;L%;M%;29,H%;K%;
 1090 OSCLI"FX11,20":OSCLI"FX4,1"
 1100 ENDPROC
 1110 :
 1120 DEF PROCgrid
 1130 FORI%=0TOR%:MOVE0,I%*E%:DRAWL%-H%,
I%*E%:NEXT
 1140 FORJ%=0TOQ%:MOVEJ%*D%,0:DRAWJ%*D%,
M%-K%:NEXT
 1150 VDU7,31,0,2:PRINT"Which character
do you wish to redefine?":*FX15
 1160 REPEAT:Z%=GET:UNTILZ%>31ANDZ%<127
 1170 ENDPROC
 1180 :
 1190 DEF PROCinfo
 1200 COLOUR1:VDU31,8,0:PRINTA$;"Charact
er Designer":FORI%=24TO26:PROCclear(I%):
NEXT
 1210 VDU31,0,24:COLOUR2:PRINT"The Space
 Bar sets a point"'"The Delete key clear
s a point"'"Use the arrow keys to move t
he cursor"
 1220 VDU31,0,2:PRINT"Press RETURN when
you have finished";SPC5
 1230 VDU31,13,6:PRINT"Redefining ";:COL
OUR1:PRINTCHR$Z%
 1240 VDU31,0,28:PRINT"Grid co-ordinates
:";SPC10;:VDU31,30,28:PRINT"Flag:"
 1250 ENDPROC
 1260 :
 1270 DEF PROCdesign
 1280 REPEAT:PROCkeys:UNTILG%=13ORG%=81
 1290 IFG%=13 AND N%=0 PROCcalc1
 1300 IFG%=13 AND N%=1 PROCcalc2
 1310 IFG%=81 THEN V%=TRUE:PROCclear(2)
 1320 ENDPROC
 1330 :
 1340 DEF PROCclear(J%)
 1350 VDU31,0,J%:PRINTSPC40
 1360 ENDPROC
 1370 :
 1380 DEF PROCkeys
 1390 PROCcursor
 1400 REPEAT:VDU31,20,28:PRINT;X%;SPC2;Y
%;SPC3:VDU31,35,28:PRINTgrid%(X%,Y%)
 1410 G%=GET:UNTILG%=13ORG%=32ORG%=81ORG
%=127OR(G%>135ANDG%<140):PROCcursor
 1420 IF G%=136 X%=X%-1:IF X%=0 X%=Q%
 1430 IF G%=137 X%=X%+1:IF X%=Q%+1 X%=1
 1440 IF G%=138 Y%=Y%-1:IF Y%=0 Y%=R%
 1450 IF G%=139 Y%=Y%+1:IF Y%=R%+1 Y%=1
 1460 IF G%=32 grid%(X%,Y%)=1:GCOL0,1:PR
OCshow(3)
 1470 IF G%=127 grid%(X%,Y%)=0:GCOL0,1:P
ROCshow(0):GCOL0,3:PROCbox
 1480 ENDPROC
 1490 :
 1500 DEF PROCbox
 1510 MOVE(X%-1)*D%,(Y%-1)*E%:DRAWX%*D%,(Y%-1)*E%:DRAWX%*D%,Y%*E%:DRAW(X%-1)*D%,Y%*E%:DRAW(X%-1)*D%,(Y%-1)*E%
 1520 ENDPROC
 1530 :
 1540 DEF PROCcursor
 1550 GCOL3,2:PROCbox
 1560 ENDPROC
 1570 :
 1580 DEF PROCshow(C%)
 1590 PROCbox:GCOL0,C%:PLOT85,X%*D%,Y%*E%:PLOT85,X%*D%,(Y%-1)*E%
 1600 ENDPROC
 1610 :
 1620 DEF PROCcalc1
 1630 PROCmessage:PROCwipe
 1640 FOR I%=1 TO Q%:FOR J%=1 TO R%
 1650 bytes%(I%)=bytes%(I%)+grid%(I%,J%)*(2^(J%-1)):NEXT:NEXT
 1660 PROCbytes
 1670 ENDPROC
 1680 :
 1690 DEF PROCcalc2
 1700 PROCmessage:PROCwipe
 1710 FORI%=1TOQ%*(N%+1)STEP2:FORJ%=1TO8:bytes%(I%)=bytes%(I%)+grid%(I%DIV2+1,J%+8)*(2^(J%-1)):NEXT:NEXT
 1720 FORI%=2TOQ%*(N%+1)STEP2:FORJ%=1TO8:bytes%(I%)=bytes%(I%)+grid%(I%DIV2,J%)*(2^(J%-1)):NEXT:NEXT
 1730 PROCbytes
 1740 ENDPROC
 1750 :
 1760 DEF PROCmessage
 1770 PROCclear(0):PROCclear(2):FORI%=24TO28:PROCclear(I%):NEXT
 1780 VDU31,0,0:PRINT"Please wait...Calculating."
 1790 ENDPROC
 1800 :
 1810 DEF PROCwipe
 1820 FORI%=1TOQ%*(N%+1):bytes%(I%)=0:NEXT
 1830 ENDPROC
 1840 :
 1850 DEF PROCbytes
 1860 S%=0:REPEAT:S%=S%+1:A%=bytes%(S%)
 1870 UNTIL A%ORS%=(-8*(N%=0)-28*(N%=1))
 1880 F%=(S%+(N%=0))DIV(1-3*(N%=1))
 1890 S%=Q%*(N%+1)+1:REPEAT:S%=S%-1:A%=bytes%(S%):UNTIL A% OR S%=0
 1900 W%=(S%-(N%=1))DIV(1-3*(N%=1))
 1910 IF W%<F% PROCmistake:ENDPROC
 1920 IFW%-F%<5 AND F%<7 W%=F%+5
 1930 PROCclear(0):VDU7,31,0,0:COLOUR2:PRINT"Upper or lower 8 pins? ";:COLOUR1:PRINT"(U/L)":*FX15
 1940 REPEAT:G%=INSTR("UuLl",GET$):UNTIL G%
 1950 IFG%<=2 byte%=&80 ELSEbyte%=0
 1960 VDU7,31,0,0:COLOUR2:PRINT"Do you wish to use the full width? ";:COLOUR1:PRINT"(Y/N)":*FX15
 1970 REPEAT:T%=INSTR("YyNn",GET$):UNTIL T%
 1980 IFT%<=2 F%=0:W%=11
 1990 attrib%=byte%ORF%*&10ORW%:PROCclear(0):VDU31,0,24:COLOUR2:PRINT"Data attribute byte: ";:COLOUR1:PRINTattrib%
 2000 VDU31,0,25:COLOUR2:PRINT"Data bytes: ";:COLOUR1:FORI%=1TOQ%*(N%+1)
 2010 PRINTbytes%(I%);:NEXT
 2020 ENDPROC
 2030 :
 2040 DEF PROCmistake
 2050 PROCclear(0):VDU7,31,0,0:PRINT"Error! No character defined!"
 2060 V%=TRUE:VDU31,0,2:PRINT"Press SPACE to continue":*FX15
 2070 REPEAT UNTIL GET=32
 2080 ENDPROC
 2090 :
 2100 DEF PROCprint
 2110 IFV%=TRUE THEN ENDPROC
 2120 VDU7,31,0,0:COLOUR2:PRINT"Send data to printer? ";:COLOUR1:PRINT"(Y/N)":*FX15
 2130 REPEAT:T%=INSTR("YyNn",GET$):UNTIL T%
 2140 IFT%<=2 PROCprinter
 2150 ENDPROC
 2160 :
 2170 DEF PROCprinter
 2180 IF N%=0 THEN U%=0 ELSE U%=2
 2190 PROCclear(0):VDU7,31,0,0:PRINT"Ple
ase check the printer is on line"
 2200 VDU31,0,2:PRINT"Press SPACE to sen
d data to printer":*FX15
 2210 REPEAT UNTIL GET=32
 2220 VDU2,1,27,1,108,1,9,1,27,1,81,1,69
,1,13:REM Set left and right margins and
 carriage return
 2230 VDU21:PRINT"Data for ";CHR$Z%
 2240 PRINT 27,38,U%,Z%,Z%;:REM Control
Codes
 2250 PRINTattrib%;:FORJ%=1TOQ%*(N%+1):P
RINTbytes%(J%);:NEXT:PRINT''A$+"characte
r: ";
 2260 VDU6,1,27,1,38,1,U%,1,Z%,1,Z%,1,at
trib%;:REM Redefinition codes
 2270 FORJ%=1TOQ%*(N%+1):VDU1,bytes%(J%)
;:NEXT:REM Data
 2280 VDU1,27,1,37,1,1;:REM Download RAM
 2290 VDU1,Z%,1,32;:REM Print character
 2300 VDU1,27,1,119,1,1,1,27,1,87,1,1;:R
EM Double height and width
 2310 VDU1,Z%,1,32;:REM Large character
 2320 VDU1,27,1,119,1,0,1,27,1,87,1,0;:R
EM Normal height and width
 2330 VDU1,27,1,120,1,1;:REM Select NLQ
 2340 VDU1,Z%,1,32;:REM Print character
 2350 VDU1,27,1,119,1,1,1,27,1,87,1,1;:R
EM Double height and width
 2360 VDU1,Z%;:REM Double size character
 2370 VDU1,27,1,119,1,0,1,27,1,87,1,0;:R
EM Normal height and width
 2380 VDU1,27,1,120,1,0;:REM Draft
 2390 VDU1,27,1,37,1,0;:REM ROM font
 2400  VDU1,27,1,108,1,0,1,27,1,81,1,80,
1,13:REM Reselect original margin positi
ons and carriage return
 2410 TIME=0:REPEAT UNTIL TIME>200:VDU3
 2420 FORJ%=0TO6:PROCclear(J%):NEXT
 2430 VDU7,31,0,2:COLOUR2:PRINT"Printout
 of grid pattern?  ";:COLOUR1:PRINT"(Y/N
)":*FX15
 2440 REPEAT:T%=INSTR("YyNn",GET$):UNTIL
T%
 2450 IFT%<=2 PROCprintgrid
 2460 PROCclear(2):VDU7,31,0,2:PRINT"Pre
ss SPACE to continue":*FX15
```

```
 2470 REPEAT UNTIL GET=32
 2480 ENDPROC
 2490 :
 2500 DEF PROCprintgrid
 2510 @%=&01:RESTORE2740
 2520 VDU2:FORI%=1TO40:READcode%
 2530 VDU1,code%;:NEXT:VDU21
 2540 FORJ%=R%TO1STEP-1
 2550 VDU124;:REM Vertical bar
 2560 FORI%=1TOQ%:PRINTgrid%(I%,J%);:NEX
T
 2570 VDU124,13:REM Vertical bar + C.R.
 2580 NEXT:PRINT:VDU6
 2590 RESTORE2800:FORI%=1TO13:READcode%
 2600 VDU1,code%;:NEXT
 2610 TIME=0:REPEAT UNTIL TIME>200
 2620 VDU3,7:@%=&04
 2630 ENDPROC
 2640 :
 2650 DEF PROCchoose
 2660 FORI%=0TO6STEP2:PROCclear(I%):NEXT
:FORI%=23TO29:PROCclear(I%):NEXT
 2670 VDU31,0,24:COLOUR2:PRINT"Edit ";CH
R$Z%;" further?":VDU31,27,24:COLOUR1:PRI
NT"1"
 2680 VDU31,0,26:COLOUR2:PRINT"Define an
other character?  ";:COLOUR1:PRINT"2"
 2690 VDU7,31,0,28:COLOUR2:PRINT"Quit pr
ogram?":VDU31,27,28:COLOUR1:PRINT"3":*FX
15
 2700 REPEAT:G%=GET:UNTILG%>48ANDG%<52
 2710 IFG%=49 V%=FALSE ELSE V%=TRUE
 2720 ENDPROC
 2730 :
 2740 DATA 27,58,0,0,0:REM ROM to RAM
 2750 DATA 27,38,0,48,49:REM Define 0+1
 2760 DATA 251,0,0,0,0,0,0,0,0,0,0,0:REM
 Data for 0
 2770 DATA 139,0,60,0,126,0,255,0,126,0,
60,0:REM Data for 1
 2780 DATA 27,37,1:REM Download font
 2790 DATA 27,108,25:REM Left margin 25
 2800 DATA 27,37,0:REM Select ROM font
 2810 DATA 27,58,0,0,0:REM ROM to RAM
 2820 DATA 27,108,0,13,13:REM Left margi
n 0 and 2 carriage returns
```

Emblems

Alan Gray presents a game of luck, skill and jolly little pictures.

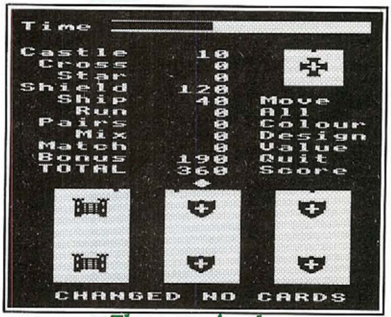
Emblems is a Yahtzee style game in which you are dealt a succession of nine hands of three cards and a trump card, rather than rolling a set of five dice. In each hand you are allowed up to four attempts to alter any of the three cards, but the trump card cannot be altered.

The emblems are like suits in playing cards, but there are five different types: castles, crosses, stars, shields or ships, which may be red, blue or black. The design, colour and number of emblems (one to three) on each card are chosen at random at the beginning of each hand.

The object of the game is to try to alter each hand, in as few attempts as possible, so that it matches the requirements of one of the nine scoring boxes. In most cases this means trying to end up with the same three emblems in the same colour on all three cards. Often you receive a bonus if the design and/or colour of the trump card is the same as that of your scoring hand.

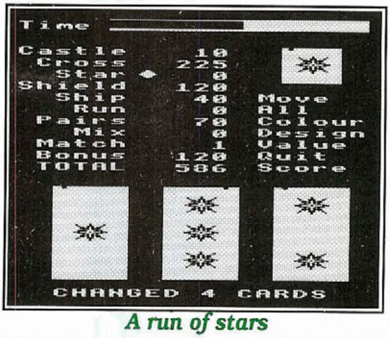
WORTH THE EFFORT

The program listed at the end should be typed in. As it's long you should save at regular intervals - keep at it, the game is well worth the effort!



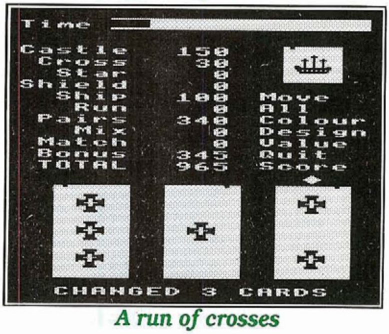
```
Time
Castle     10
 Cross      0
  Star      0
Shield    120
  Ship     40   Move
   Run      0   All
 Pairs      0   Colour
   Mix      0   Design
 Match      0   Value
 Bonus    190   Quit
 TOTAL    360   Score

CHANGED NO CARDS
```

The game in play



```
Time
Castle     10
 Cross    225
  Star      0
Shield    120
  Ship     40   Move
   Run      0   All
 Pairs     70   Colour
   Mix      0   Design
 Match      1   Value
 Bonus    120   Quit
 TOTAL    586   Score

CHANGED 4 CARDS
```

A run of stars



```
Time
Castle    150
 Cross     30
  Star      0
Shield      0
  Ship    100   Move
   Run      0   All
 Pairs    340   Colour
   Mix      0   Design
 Match      0   Value
 Bonus    345   Quit
 TOTAL    965   Score

CHANGED 3 CARDS
```

A run of crosses

At the start of each game select both the level of play and sound effects. Try to improve the scoring potential of each hand by altering some or all of the three cards.

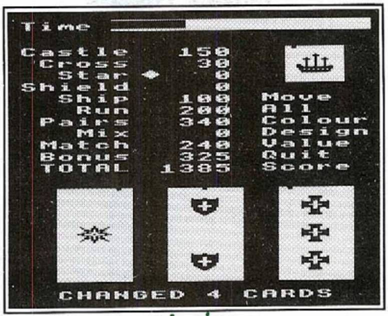
When satisfied with the hand you quit by pressing 'Q', and score the hand by moving a pointer to the appropriate box on the score card then pressing 'S'. The score will be entered, and the total will be updated.

This cycle is repeated until the nine boxes on the score card are filled (each can only be used once); any time bonuses are added to this total. High scores are kept track of on the high score table.

CARD CHANGES

Press 'M' to move the pointer to one of the cards, and press 'C' if you want to alter the colour, press 'D' to alter the design or press 'V' to change the value (number of emblems) on the card. If you press 'A' then all three things will be changed simultaneously. For example if a card has one red castle, pressing 'A' might change it to two black crosses. Every time you alter a card, five points are deducted from your starting bonus of 250 points.

When you are happy with the hand press 'Q' to quit; the pointer moves to the scorecard. After four changes, the pointer automatically moves to the scorecard.



```
Time
Castle      150
 Cross       30
  Star ◆      0
Shield        0
  Ship      100   Move
   Run      200   All
 Pairs      340   Colour
   Mix        0   Design
 Match      240   Value
 Bonus      325   Quit
 TOTAL     1385   Score

CHANGED 4 CARDS
```

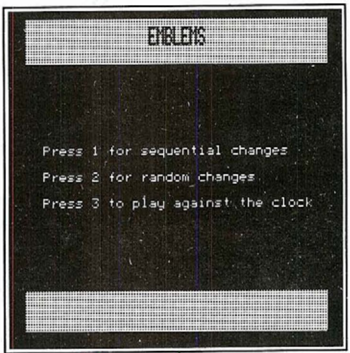
A mix

SCORING COMBINATIONS

The score you get in each box is worked out as follows:

Castle, Cross, Star, Shield and **Ship** - You will score 25 points for each emblem in your hand that matches the box you have chosen.

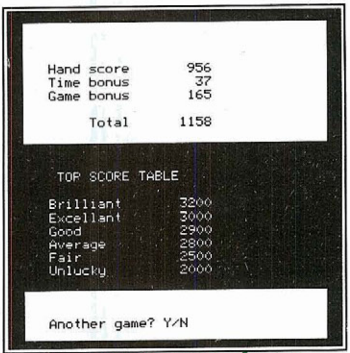
Run - A run is formed when each card in the hand has a different value, e.g. You could have two stars, one ship and three castles, which would score 40. But, a run where all emblems are the same, scores 200 points.



```
EMBLEMS

Press 1 for sequential changes

Press 2 for random changes

Press 3 to play against the clock
```

The opening menu

Pairs - You score 50 points for each pair of cards with the same design, and 20 for each pair with the same value (but different design). For example, a hand with two stars, three stars and three stars, would score 150 points for three pairs of stars plus 20 points for a pair of threes.



```
Hand score       956
Time bonus        37
Game bonus       165

     Total      1158

 TOP SCORE TABLE

Brilliant       3200
Excellant       3000
Good            2900
Average         2800
Fair            2500
Unlucky         2000

Another game? Y/N
```

How not to get on the top score table

In the first seven boxes, extra points can be earned as follows: if all three cards are the same colour the score in that hand is doubled. If all three cards and the trump card have the same design, 50 points are added to the bonus box. If all three cards and the trump card are the same colour, 20 points are added to the bonus box.

Mix - Here you score most points when each card is different in colour, design and value and even more points are scored when the design on the trump card is different from each card forming the mix.

Match - Each emblem in your hand which matches the one displayed on the trump card in colour and design scores 80 points, but the same design with a different colour scores 20 points.

SCORING

On pressing 'Q', or after you have made four changes, the pointer moves to the first unused box on the scorecard. Press 'M' to move the pointer to the box which best matches your hand. Press 'S' to enter the score, if the hand doesn't match the requirements of the box a single point is entered to show that the box has been used.

LEVELS OF PLAY

At level one the effect of card changes are predictable because they are sequential, so that colour changes from blue to red to black, then blue again. Likewise designs change from castle to cross to star to shield to ship and back to castle, values go one to two to three to one etc. You have 15 minutes to complete the game, the elapsed time is indicated by the 'fuel gauge' at the top of the screen.

Every unused second scores a time bonus of 0.1 points.

At level two card changes are random, so you cannot predict the outcome of the hands, but the scores are doubled to compensate, otherwise play is the same as level one.

Level three is the same as level one, but you have five minutes to finish the game; each unused second scores a bonus of three points.

Take care with this game, it's one of those that will keep you up all night if you're not careful!

```
   10 REM Program Emblems
   20 REM Version B 1.0
   30 REM Author  Alan Gray
   40 REM BEEBUG  Aug/Sep 1992
   50 REM Program subject to copyright
   60 :
  100 MODE 7
  110 VDU23,224,168,248,112,80,95,90,127
,122
  120 VDU23,225,21,31,14,10,250,90,254,9
4
  130 VDU23,226,127,90,95,90,127,122,112
,112
  140 VDU23,227,254,90,250,90,254,94,14,
14
  150 VDU23,228,7,7,3,3,34,62,62,56
  160 VDU23,229,224,224,192,192,68,124,1
24,28
  170 VDU23,230,62,62,34,3,3,3,7,7
  180 VDU23,231,124,124,68,192,192,192,2
24,224
  190 VDU23,232,0,32,17,13,15,5,30,112
  200 VDU23,233,128,130,196,88,120,80,60
,135
  210 VDU23,234,30,5,15,13,17,32,0,0
  220 VDU23,235,60,80,120,88,196,130,128
,0
  230 VDU23,236,0,33,63,63,62,62,62,56
  240 VDU23,237,0,132,252,252,124,124,12
4,28
  250 VDU23,238,56,30,30,30,15,7,3,1
  260 VDU23,239,28,120,120,120,240,224,1
92,128
  270 VDU23,240,0,1,0,8,28,8,8,8
  280 VDU23,241,128,192,128,136,156,136,
136,136
  290 VDU23,242,8,200,232,255,127,106,63
,63
  300 VDU23,243,136,136,139,255,255,170,
254,252
  310 VDU23,244,24,24,24,0,0,0,0,0
  320 VDU23,245,255,255,255,255,153,153,
255,255
  330 VDU23,246,255,255,102,102,126,60,6
0,0
  340 VDU23,247,0,0,102,0,0,36,0,24
  350 VDU23,248,28,30,94,94,126,62,28,28
  360 VDU23,252,8,8,8,73,107,62,28,8
  370 VDU23,253,1,3,7,15,7,3,1,0
  380 VDU23,254,128,192,224,240,224,192,
128,0
  390 VDU23,255,255,255,255,255,255,255,
255,255
  400 PROCinit
  410 REPEAT
  420 MODE7
  430 bbk$=CHR$132+CHR$157+CHR$131
  440 rbk$=CHR$129+CHR$157+CHR$135
  450 PROCscr
  460 PRINTTAB(15,1)CHR$141;"EMBLEMS"
  470 PRINTTAB(15,2)CHR$141;"EMBLEMS"
  480 PRINTTAB(3,10)"Press 1 for sequent
ial changes"
  490 PRINTTAB(3,12)"Press 2 for random
changes"
  500 PRINTTAB(3,14)"Press 3 to play aga
inst the clock"
  510 REPEAT: game%=GET-48
  520 UNTIL game%>0 AND game%<4
  530 IF game%=3 T%=300:tf%=3 ELSE T%=90
0:tf%=1
  540 PRINTTAB(3,22)"PRESS H L or O for
SOUND Hi Lo Off"
  550 S%=0
  560 REPEAT: s$=GET$
  570 UNTIL s$="H" OR s$="L" OR s$="O"
  580 IF s$="H" S%=-15
  590 IF s$="L" S%=-8
  600 MODE 5
  610 VDU19,0,4;0;19,2,0;0;19,3,3;0;
  620 IF S%<>0 SOUND1,1,220,2
  630 FOR J%=1 TO 9:SC%(J%)=0:NEXT
  640 FOR J%=10 TO 11:SC%(J%)=250:NEXT
  650 CLS:PROCtitle:numb%=0:TIME=0
  660 GCOL0,3
  670 FOR J%=960 TO 996 STEP4
  680 MOVE320,J%:DRAW1220,J%
```

```
 690 NEXT
 700 REPEAT
 710 CH%=0:numb%=numb%+1
 720 VDU4,17,3
 730 PROCtime
 740 PRINTTAB(2,30);"CHANGED NO CARDS"
 750 PROCtable:PROCmaster
 760 FOR J%=1 TO 3
 770 Q$="A":PROCrnd:PROCemblems
 780 NEXT
 790 W=INKEY(1)
 800 IF S%<>0 SOUND1,S%,189,5
 810 VDU5:PROCchoose:GCOL0,2:VDU4
 820 PROCscore
 830 UNTIL numb%=9
 840 W$=INKEY$(10)
 850 MODE7:PROCtoptbl
 860 UNTIL FALSE
 870 END
 880 :
1000 DEF PROCinit
1010 VDU23;8202;0;0;0;19,0,4;0;19,3,3;0
;
1020 ENVELOPE1,7,2,1,1,1,1,1,121,-10,-5
,-2,80,120
1030 DIM C%(3),D%(3),V%(3),B%(4)
1040 DIM SC%(11),A$(5),N$(18)
1050 DIM P%(7),P$(7)
1060 FOR J%=1 TO 7
1070 READ P$(J%),P%(J%)
1080 NEXT
1090 J%=223
1100 FOR K%=1 TO 5
1110 A$(K%)=CHR$(J%+1)+CHR$(J%+2)+CHR$1
0+CHR$8+CHR$8+CHR$(J%+3)+CHR$(J%+4)
1120 J%=J%+4
1130 NEXT
1140 B$=CHR$253+CHR$254
1150 N$(1)="Castle":N$(2)=" Cross"
1160 N$(3)="  Star":N$(4)="Shield"
1170 N$(5)="  Ship":N$(6)="   Run"
1180 N$(7)=" Pairs":N$(8)="   Mix"
1190 N$(9)=" Match":N$(10)=" Bonus"
1200 N$(11)=" TOTAL":N$(12)="Move"
1210 N$(13)="All":N$(14)="Colour"
1220 N$(15)="Design":N$(16)="Value"
1230 N$(17)="Quit":N$(18)="Score"
1240 FOR J%=1 TO 5:C$=C$+CHR$255:NEXT
1250 ENDPROC
1260 :
1270 DEF PROCscr
1280 FOR j%=0 TO 3:PRINTrbk$:NEXT
1290 FOR j%=4 TO 20:PRINTbbk$:NEXT
```

```
1300 FOR j%=21 TO 23:PRINTrbk$:NEXT
1310 ENDPROC
1320 :
1330 DEF PROCtime
1340 tl%=TIME DIV 100:tu%=tl%*tf%
1350 IF tu%>900 tu%=900
1360 GCOL0,1
1370 FOR J%=320 TO 320+tu% STEP 8
1380 MOVE J%,968:DRAWJ%,992
1390 NEXT
1400 IF tl%>=T% THEN numb%=9
1410 PRINTTAB(0,1);"Time";
1420 ENDPROC
1430 :
1440 DEF PROCrnd
1450 GCOL0,3:VDU29,J%*384-128;256;
1460 MOVE-128,-160:MOVE128,-160
1470 PLOT85,128,160:MOVE128,160
1480 MOVE-128,160:PLOT85,-128,-160
1490 IF Q$="A" D%(J%)=RND(5):C%(J%)=RND
(3)-1:V%(J%)=RND(3):ENDPROC
1500 IF Q$="C" C%(J%)=RND(3)-1:ENDPROC
1510 IF Q$="V" V%(J%)=RND(3):ENDPROC
1520 IF Q$="D" D%(J%)=RND(5):ENDPROC
1530 ENDPROC
1540 :
1550 DEF PROCchange
1560 VDU18,0,3,29,J%*384-128;256;
1570 MOVE-128,-160:MOVE128,-160
1580 PLOT85,128,160:MOVE128,160
1590 MOVE-128,160:PLOT85,-128,-160
1600 IF Q$="C" C%(J%)=C%(J%)+1:IF C%(J%
)>2 C%(J%)=0
1610 IF Q$="D" D%(J%)=D%(J%)+1:IF D%(J%
)>5 D%(J%)=1
1620 IF Q$="V" V%(J%)=V%(J%)+1:IF V%(J%
)>3 V%(J%)=1
1630 IF Q$<>"A" ENDPROC
1640 C%(J%)=C%(J%)+1:IF C%(J%)>2 C%(J%)
=0
1650 D%(J%)=D%(J%)+1:IF D%(J%)>5 D%(J%)
=1
1660 V%(J%)=V%(J%)+1:IF V%(J%)>3 V%(J%)
=1
1670 ENDPROC
1680 :
1690 DEF PROCemblems
1700 VDU29,J%*384-192;288;18,0,0
1710 MOVE(C%(J%)*90-54),132
1720 PRINTCHR$244
1730 FOR K%=0 TO 96 STEP 16
1740 GCOL0,C%(J%):MOVE0,K%:PRINTB$
1750 MOVE0,-K%:PRINTB$
```

```
 1760 GCOL0,3:MOVE0,K%:PRINTB$
 1770 MOVE0,-K%:PRINTB$
 1780 NEXT
 1790 GCOL0,C%(J%)
 1800 IF V%(J%)=1 OR V%(J%)=3 PROCpos(0)
 1810 IF V%(J%)=2 OR V%(J%)=3 PROCpos(-9
6):PROCpos(96)
 1820 VDU29,0;0;:SOUND1,S%,149+J%*24,5
 1830 ENDPROC
 1840 :
 1850 DEF PROCpos(Y%)
 1860 MOVE0,Y%:PRINTA$(D%(J%))
 1870 ENDPROC
 1880 DEF PROCchoose
 1890 CH%=0:J%=1
 1900 REPEAT
 1910 VDU29,0;0;
 1920 PROCptr(3)
 1930 REPEAT
 1940 *FX15,1
 1950 Q$=GET$
 1960 IF Q$="M" THEN PROCptr(0):PROCmove
ptr
 1970 UNTIL Q$<>"M"
 1980 IF Q$="A" OR Q$="C" OR Q$="D" OR Q
$="V" THEN good=-1 ELSE good=0
 1990 IF good AND (game%=1 OR game%=3) T
HEN PROCchange:PROCemblems:PROCupdate
 2000 IF good AND game%=2 THEN PROCrnd:P
ROCemblems:PROCupdate
 2010 UNTIL CH%=4 OR Q$="Q"
 2020 PROCptr(0)
 2030 SOUND 1,S%,200,10:W=INKEY(5)
 2040 ENDPROC
 2050 :
 2060 DEF PROCupdate
 2070 CH%=CH%+1
 2080 IF SC%(10)>0 THEN FOR SB=10 TO 11:
SC%(SB)=SC%(SB)-5:NEXT
 2090 VDU4,17,3
 2100 PRINTTAB(10,30);SPC(8);TAB(10,30);
CH%;" CARDS"
 2110 VDU5:FOR SB%=10 TO 11:GCOL0,0
 2120 MOVE416,936-SB%*40:PRINTC$
 2130 GCOL0,3
 2140 MOVE96,936-SB%*40:PRINTSC%(SB%)
 2150 NEXT
 2160 ENDPROC
 2170 :
 2180 DEF PROCscore
 2190 VDU5,29,0;0;18,3,3:L%=0:bon%=0
 2200 REPEAT
 2210 REPEAT
```

```
 2220 L%=L%+1:IF L%=10 L%=1
 2230 UNTIL SC%(L%)=0
 2240 MOVE 392,940-L%*40:PRINT;B$
 2250 *FX15,0
 2260 Q$=GET$
 2270 MOVE 392,940-L%*40:PRINT;B$
 2280 SOUND 1,S%,L%*8+65,2
 2290 W=INKEY(5)
 2300 UNTIL Q$="S"
 2310 PROCscbrd
 2320 ENDPROC
 2330 :
 2340 DEF PROCscbrd
 2350 IF L%<6 PROCembs:PROCmatcol
 2360 IF L%=6 PROCrun:PROCmatcol
 2370 IF L%=7 PROCpair:PROCmatcol
 2380 IF L%=8 PROCmix
 2390 IF L%=9 PROCmatch
 2400 IF MC%=C%(1) AND MC%=C%(2) AND MC%
=C%(3) bon%=bon%+20
 2410 IF MD%=D%(1) AND MD%=D%(2) AND MD%
=D%(3) bon%=bon%+50
 2420 SC%(10)=SC%(10)+bon%
 2430 IF SC%(L%)=0 SC%(L%)=1
 2440 IF game%=2 SC%(L%)=SC%(L%)*2
 2450 GCOL0,0:MOVE416,936-L%*40:PRINTC$
 2460 GCOL0,3:MOVE96,936-L%*40:PRINTSC%(
L%)
 2470 IF S%<>0 SOUND1,1,220,70
 2480 SC%(11)=SC%(11)+SC%(L%)+bon%
 2490 FOR SB%=10 TO 11:GCOL0,0
 2500 MOVE416,936-SB%*40:PRINTC$:GCOL0,3
 2510 MOVE96,936-SB%*40:PRINTSC%(SB%):NE
XT
 2520 VDU5
 2530 IF numb%=9 W=INKEY(500)
 2540 ENDPROC
 2550 :
 2560 DEF PROCmatcol
 2570 IF C%(1)=C%(2) AND C%(1)=C%(3) SC%
(L%)=SC%(L%)*2
 2580 ENDPROC
 2590 :
 2600 DEF PROCtable:VDU5,18,0,2
 2610 *FX15,0
 2620 FOR X%=904 TO 464 STEP-40
 2630 MOVE 400 ,X%:DRAW 764,X%:NEXT
 2640 FOR X%=400 TO 764 STEP 364
 2650 MOVE X%,464:DRAW X%,904:NEXT
 2660 FOR L%=1 TO 11:GCOL0,0
 2670 MOVE416,936-L%*40:PRINTC$:GCOL0,3
 2680 MOVE96,936-L%*40:PRINTSC%(L%)
```

continued on page 57

Free-Text Datafiles

Gil Swain muses on the idea of a totally flexible text database.

In the March Postbag (Vol.10, No.9) there was a letter from a reader describing his need for a database and looking for guidance on which commercial products would be most suitable. Although certain products were recommended, it occurred to me that one of the requirements could best be met by a different solution.

The whole point about information is that it does not always fall into neatly definable structures but, with a little thought, a structure of sorts can be imposed.

The following can be implemented with nothing more complicated than a word processor like Wordwise or even a simple text editor, providing it allows for block copying, block moving, and character, word or phrase searching. The essential idea is to find a way of representing information without duplicating the original and with minimal outlay in terms of time and effort.

Free-Text is a piece of jargon used to describe the storage of data in irregular form. It is 'free' because there are no constraints placed on how little or how much detail is held on file. In its widest sense any piece of written or printed information can be regarded as a data file. However, it would be extremely wasteful to insert whole articles, letters etc. into a database since all that is required is an indication of where data exists and some pointers as to content.

Faced with a need to organise a mass of irregular information, we really have only three options:

1) Use a conventional database program. This will list titles, authors, location pointers, and may even identify some of the content.

2) Use a cross-referencing database (often erroneously called relational). This may go further towards organising the data, but requires more user input.

3) Create a free text system which can evolve with use.

Enough has been written in BEEBUG and elsewhere about the first two so I propose to deal only with the third. I will use as a working example an article from an earlier edition of BEEBUG. The article is entitled *3D Landscapes* by John Greening (BEEBUG Vol.7 No.9).

In a conventional database the article might be described as *Computer Generated Landscapes*. In our worked example, although we will include the title, the author and its location within BEEBUG, we will seek to represent the article content. Here is the first paragraph.

> *The program presented here draws three dimensional landscapes. Each time you run the program you are likely to get a different picture because of the use of random numbers. Moreover you will be able to influence the final outcome by controlling parameters in one or two of the procedures. The number of possible countrysides you can draw is very large indeed.*

The following words and phrases strike me as significant: *three dimensional* *landscapes, each time, different picture, random numbers, influence, outcome, controlling parameters.*

In my opinion the selected words define the subject of the paragraph and they will be used as the basis of an entry into our free-text database. The words can be left as listed, or perhaps rewritten as follows:

This program produces 3D landscapes based on random numbers and additionally by controlling selected parameters.

So far we have only looked at one paragraph, fortunately the author put a lot of detail in his opening paragraph and, on this occasion, it almost completely defines the whole article. All that it seems necessary to add is that the author provides two different versions of his program: one for the model B and another for the Master and/or enhanced model B.

The guiding principle is not to give full chapter and verse, only to convey an idea of the article content. Our final example now reads:

3D landscapes by John Greening program producing 3D Landscapes based on Random Numbers and by control of selected parameters. Versions for Model B & for BBC Master and/or Enhanced Model B.
BEEBUG March 1989

Now in best TV Cookshow tradition here are some I prepared earlier:

Julia Sets by John Greening iterative calculations in the complex plane, Gaston Julia, use of complex numbers, relation with Mandelbrot Sets program requires user input, for Model B with notes on modifying program for BBC Master.
BEEBUG May 1989

Shaping Up The Galaxies by Mike Cook Fractal generation, Diffusion Limited Aggregation, Computer modelling, micro simulation Program with notes on varying results.
Micro User July 1989

A3000 Mandelbrot Reply by Kenn Garroch to a reader's letter how to write a 'Basic' Mandelbrot program.
Popular Computing September 28 1989

Those fascinating fractals - Mike Cook deterministic fractals generated by random numbers, Strange Attractor, Geometrical Transformation, Sierpinsky Triangle, Hilbert Curves.
Micro User November 1989

To the above we can now add our worked example. Admittedly the file does not appear to have any structure, but using the facilities of the word processor we can re-order the extracts (paragraphs) in any sequence we choose. Alternatively, we can search the whole file for characters, words or phrases.

By now it has probably occurred to the reader that free-text may seem suited to five or fifty records, but what about five hundred or five thousand records. The answer is to evolve a hierarchy of free-text files. If we consider the five examples above it becomes possible to copy each extract into one or more datafiles based on author names, magazine locations or even by article content. As an example John Greening's articles could be in one datafile, Mike Cook's in another. Having decided on the number of separate datafiles and the rationale for each, that detail can now be stored in a master datafile.

continued on page 24

Mr Toad's Macro ROM

A ribbiting ROM utility to save typing.

There's no denying it, toads are lazy creatures. They make excellent programmers; in fact, they are the most intelligent life-forms in this spiral arm of the galaxy - so cunning are they that they deliberately fail all intelligence tests in order to avoid being made to work. It is therefore not surprising that Mr T has come up with that rarity, a computer program which actually saves work.

I help to run a charity called PMS Help and answer a lot of enquiries using View. Most of these involve several standard phrases such as "3-hourly carbohydrate diet", "flour, oats, rye, potatoes or rice" and "keep the chart for 2-3 months". After struggling for a while with View's macro facility I gave up and wrote one of my own that should work with any word processor, though you must have a machine with sideways RAM. With Mr Toad's Macro ROM you can type in up to ten strings, each up to 31 characters, and then insert them into the input stream at will by pressing Ctrl and the appropriate number. Better still, you can arrange for a !BOOT file to load the ROM image from disc and feed in a set of macros automatically. Nowadays I just insert the appropriate disc and boot it - up comes View in the right mode and so on, and the macros are set up with my standard phrases so that Ctrl-1, for example, inserts "PO Box 160, St Albans, AL1 4UQ" into the text whenever needed.

Being a sideways ROM which operates by grabbing a vector, it also works when programming in Basic (including the assembler, of course), so you can feed into a listing all sorts of commonly used statements like DEFPROC or g$=CHR$(ASC GET$ AND &DF). While it *should* work with other word processors and languages, beware of clashes due to your language ROM grabbing the same vector or whatever.

Type in the listing, saving at frequent intervals; I suggest the filename *MACrom*. Run the program and it will assemble the code, find a vacant sideways RAM slot (or tell you that none are free) and initialise the ROM. There is no need to press Ctrl-Break to initialise it; the program will do that too. Not all sideways ROM images which grab vectors need Ctrl-Break to initialise them, and it is good programming practice to avoid the necessity where possible. If a ROM also claims workspace, however, this may only be done on Ctrl-Break and the program may then also wish to grab any vectors at the same time; see my article on initialising ROM images in BEEBUG Vol.9 No.5. MACrom claims no workspace; it grabs its vector when it gets the *SM or *EM commands and releases it on the *DM or *WM commands, all described below.

To set up the macros, you use the star command *SM from the keyboard or in your !BOOT file. First you get a listing of the macros as set up at present. After initial power-up this will naturally be ten blanks. You are invited to press the numeric key to be programmed, or 'Q' or Escape to quit. Touch a key 0 - 9 and you will be invited to type in the text for the associated macro, terminated by Return. If you try to exceed the limit of 31 characters the string will terminate automatically. You can feed in the texts of the macros in any order. If you make an error, correct it by inserting the right text - Delete doesn't work. The cycle of *number-text-number-text* will go round and round until 'Q' or Escape is pressed in place of a number.

GETTING THE BOOT

How can you achieve this from within a !BOOT file? Well, !BOOT or any other file which is *EXECed is inserted character by character into the input stream exactly as if it had been entered from the keyboard, just like our macros. Therefore put something like this in your !BOOT file.

```
*BASIC
CHAIN "MACrom"
*WORD
*SM
0PMS Help
1PO Box 160, St Albans, AL1 4UQ
etc .....
8PMS isn't WHAT, it's WHEN it is
9Do NOT take Vitamin B6
Q
```

Don't put a space after the number unless you wish the macro to begin with one. You can, of course, end each one with a space if required. As each number is read from the file and inserted into the keyboard buffer, MACrom will move to "Enter text of macro". The carriage return at the end of each line will terminate insertion of the macro's text and MACrom will ask for the next number. This is provided on the next line, and so on until the ROM gets the 'Q'; it then enables the macros and exits. Incidentally, I intended the way out of the setup loop to be via 'Q', rather than Escape, but since sideways ROMS hang up on Escape unless you code a response to it, I also arranged that Escape should quit the loop.

There are five star-commands provided:

*DM = 'Disable Macros'.
Turns MACrom off. Doesn't erase the texts.

*EM = 'Enable Macros'.
Turns MACrom on. Also done automatically by *SM.

*LM = 'List Macros'.
Also done automatically by *SM.

*SM = 'Set up Macros'.
See above.

*WM = 'Wipe Macros'.
Erases the texts of the macros and does *DM.

PROGRAM NOTES

For a detailed explanation of extended vectors, see the article with Mr T's Beep ROM in BEEBUG Vol.11 No.1. Briefly, a vector on the Beeb is a normally superfluous address copied into RAM by the MOS on power-up and hard break; early on in some MOS routine(s), instead of JMP from A to B we have JMP from A to vector, JMP from vector to B. All input and output operations pass through these vectors, and since they are in RAM you can change them by copying the address of your own routine into them. Your own routine is now in the chain of events mediated by that particular vector; it usually ends with a jump to the address which was in the vector before you changed it. The trouble with using this facility from sideways ROMs is that, if you're in View, Basic or whatever, your own ROM is paged out. This is because only one ROM at a time can occupy the space &8000 to &BFFF in the memory map, and Basic and View are ROMs too. What you do is leave your ROM number and the address of your routine in a place in RAM which the MOS knows about. You then divert the vector not to your

ROM, but to a certain address in the MOS which is always paged in. On receiving data passing to it, the MOS looks up your ROM number and routine address, pages your ROM in and calls it. When your ROM exits, the MOS pages back in whichever ROM was last operating.

The vector used here is INSV at &022A/B, through which pass all insertions into buffers - in this instance we are interested in the keyboard buffer. This is why we can handle only 31 characters per macro; that's the size of the keyboard buffer. One could arrange to increase the limit by feeding in longer strings in chunks of 31 bytes, but it means a lot more code, and one has to consider you, poor reader, typing in all this stuff. In any case, 31 characters is quite enough for most purposes and you can always split a long string across two or more consecutive keys.

Other vectors such as KEYV spring to mind as alternatives, but all proved to have drawbacks when tried. A lot of programming decisions are made on the basis of 'suck it and see'; unless you're a professional working to a deadline, trial-and-error is a lot of fun and even blind alleys teach you things about the machine.

The functional core of this ROM is the little bit of code at the end called *.theRealBit*. It is the address of this label which is posted for the MOS to look up when a character passes through the extended vector. When called, our routine first tests for the range 0 to 9 and characters outside these limits are passed straight on. If the character does turn out to be a numeral, we need to know if Ctrl is being held down. There is a system variable at &025A which keeps track of this sort of thing; its bit 5 will be set if Ctrl is being pressed. We therefore proceed as follows.

```
PHA \ save the numeral in case Ctrl
isn't down
LDA &025A:ORA #&40 \ the zero flag is
now set if Ctrl isn't down
BEQ exit1 \ if zero-flag set, exit via
a PLA to recover numeral
```

If we've not exited by now, we copy the macro to the address which was in the vector before we grabbed it; we do NOT send the original numeral, and we exit.

The easiest way of terminating any loop in a situation like this, where we're copying out a string of bytes, is to stick an end-marker byte onto the end of the string, (see Machine Code Corner in Vol.11 No.1) but in this instance there is no byte we could choose as end-marker which could in all circumstances safely be sent out into the input stream. A zero, for example, which does nothing when sent to OSWRCH at the end of a string to be printed, is turned into a TAB character when inserted into a View text. There are ways around this, but in the end I opted to keep a note of the length of each string and to do a countdown when copying it out, even though it involves maybe three or four more lines for you to type in. Again, the 'suck it and see' approach provided the best solution, which was entirely different from what I had first envisaged.

Once again I have used space within the sideways RAM slot containing our ROM to hold the texts of the macros and the counters. So long as you don't want to blow it into an EPROM - highly unlikely - this is the best method. ROMs like this typically use one or two kilobytes, leaving 14 or 15K free in the sideways

RAM slot; no other software is going to be able to use this space, so it is absolutely secure as well as saving space elsewhere. In any case, anyone wanting to blow an EPROM is likely to know how to change the listing to use other space in main memory, and the same goes for anyone able to chain the listings of several utility ROMs together to save slots, which is a good idea if you habitually use several such.

Finally, how can one remember which number-key controls which macro? Obviously, something like a function-key strip would be useful. If you use View or some other word-processor to compose your !BOOT file, then once it's saved, prune away lines like *SM, insert spaces between the numerals and the texts and print it out. Why not also save it under another filename? Don't forget the *LM facility, either.

Mr T hopes you find this ROM utility as useful as he does and that you have some fun with it. Please don't forget Be Nice To Reptiles Day on the 17th, and remember: you have to kiss an awful lot of frogs before you find a handsome prince, but a toad is a different kettle of fish.

```
   10 REM Program  Mr Toad's MACRO Rom
   20 REM Version  B 1.0
   30 REM Author   Mr Toad
   40 REM BEEBUG   Aug/Sep 1992
   50 REM Program subject to copyright
   60 :
  100 Z%=TOP+&0280
  110 index=&BE:indexHi=&BF
  120 insVlo=&022A:insVhi=&022B
  130 osnewl=&FFE7:osasci=&FFE3
  140 osrdch=&FFE0
  150 :
  160 FOR N%=4 TO 6 STEP 2
  170 P%=&8000:O%=Z%
  180 [ OPT N%
  190 : : : : :
```

```
  200 BRK:BRK:BRK
  210 JMP hallo
  220 EQUB &82:EQUB offset MOD &100
  230 EQUB &92
  240 .titleEtc
  250 EQUS "Mr Toad's Magic MACRO Rom"
  260 EQUW &0D00
  270 EQUS "  DM (disable macros)"
  280 EQUB &0D
  290 EQUS "  EM (enable macros)"
  300 EQUB &0D
  310 EQUS "  LM (list macros)"
  320 EQUB &0D
  330 EQUS "  SM (set up macros)"
  340 EQUB &0D
  350 EQUS "  WM (wipe out macros)"
  360 EQUW &1B0D
  370 .offset
  380 BRK:EQUS "(C) BEEBUG Aug/Sep 1992"
  390 : : : : :
  400 .print
  410 STX index:STY indexHi:LDY #&FF
  420 .loop
  430 INY:LDA (index),Y:JSR osasci
  440 CMP #&1B:BNE loop:RTS
  450 :
  460 .help
  470 LDA (&F2),Y:CMP #&0D:BNE out
  480 JSR osnewl
  490 LDX #titleEtc MOD &100
  500 LDY #titleEtc DIV &100
  510 JSR print
  520 :
  530 .out
  540 PLY:PLX:PLA:RTS
  550 :
  560 .turnItOff
  570 LDX #prompt4 MOD &100
  580 LDY #prompt4 DIV &100
  590 JSR print
  600 .tio2
  610 LDY oldVecHi:BEQ notOn
  620 LDX oldVec:SEI:STY insVhi
  630 STX insVlo:CLI:STZ oldVecHi
  640 .notOn
  650 JMP claimOut
  660 :
  670 .erase
  680 LDX #&0A:LDA #0
  690 .loop
  700 STA tallies,X
  710 DEX:BPL loop
  720 JSR list
```

```
 730 LDX #prompt5 MOD &100
 740 LDY #prompt5 DIV &100
 750 JSR print:JMP tio2
 760 :
 770 .callList
 780 JSR list:JMP claimOut
 790 :
 800 .hallo
 810 PHA:PHX:PHY
 820 CMP #9:BEQ help
 830 CMP #4:BNE out
 840 INY:INY
 850 LDA (&F2),Y:CMP #&0D:BNE out
 860 DEY:LDA (&F2),Y:AND #&DF
 870 CMP #ASC"M":BNE out
 880 DEY:LDA (&F2),Y:AND #&DF
 890 CMP #ASC"D":BEQ turnItOff
 900 CMP #ASC"E":BEQ on
 910 CMP #ASC"L":BEQ callList
 920 CMP #ASC"W":BEQ erase
 930 CMP #ASC"S":BNE out:JSR setUp
 940 :
 950 .on
 960 LDA oldVecHi:BNE listAndOut
 970 LDA insVhi:STA oldVecHi
 980 LDA insVlo:STA oldVec:LDA &F4
 990 LDY #theRealBit DIV &100
1000 LDX #theRealBit MOD &100
1010 STA &0DE0:STY &0DDF:STX &0DDE
1020 LDX #&3F:LDY #&FF
1030 SEI:STY insVhi
1040 STX insVlo:CLI
1050 .listAndOut
1060 LDX #prompt3 MOD &100
1070 LDY #prompt3 DIV &100
1080 JSR print:JSR list2
1090 :
1100 .claimOut
1110 PLY:PLX:PLA:LDA #0:RTS
1120 :
1130 .setUp
1140 JSR list
1150 LDX #prompt1 MOD &100
1160 LDY #prompt1 DIV &100
1170 JSR print
1180 .inpNum
1190 JSR osrdch:BCS claimOut
1200 ORA #&20:CMP #ASC"q":BEQ on
1210 CMP #ASC"0":BCC inpNum
1220 CMP #1+ASC"9":BCS inpNum
1230 JSR osasci:PHA
1240 LDX #prompt2 MOD &100
1250 LDY #prompt2 DIV &100
```

```
1260 JSR print:PLA:PHA
1270 JSR osasci:JSR setIndex
1280 STZ counter:JSR osnewl
1290 .loop
1300 JSR osrdch:BCS end
1310 CMP #&0D:BEQ end
1320 CMP #&20:BCC loop
1330 CMP #&7F:BCS loop
1340 JSR osasci:STA (index)
1350 INC counter:INC index
1360 LDA counter:CMP #&1F:BNE loop
1370 .end
1380 PLA:AND #&0F:TAX
1390 LDA counter:STA tallies,X
1400 JMP setUp
1410 :
1420 .list
1430 LDX #cls MOD &100
1440 LDY #cls DIV &100
1450 JSR print
1460 .list2
1470 LDA #ASC"0"
1480 .loop
1490 PHA:JSR osasci:JSR setIndex
1500 LDA #&20:JSR osasci
1510 .inner
1520 DEC counter:BMI endLine
1530 LDA (index):JSR osasci
1540 INC index:JMP inner
1550 .endLine
1560 JSR osnewl:PLA:INA
1570 CMP #1+ASC"9":BCC loop
1580 JSR osnewl:RTS
1590 :
1600 .setIndex
1610 PHX:PHA:AND #&0F
1620 TAX:LDA tallies,X
1630 STA counter:PLA
1640 ASL A:ASL A:ASL A:ASL A:ASL A
1650 STA index:LDA #0:ADC #&84
1660 STA indexHi:PLX:RTS
1670 :
1680 .theRealBit
1690 CMP #1+ASC"9":BCS mainRoutine
1700 CMP #ASC"0":BCC mainRoutine
1710 PHA:LDA &025A:AND #&40:BEQ exit
1720 PLA:PHA:JSR setIndex
1730 LDA counter:BEQ exit:PLA
1740 .loop
1750 LDA (index):INC index
1760 DEC counter:BEQ mainRoutine
1770 JSR mainRoutine:JMP loop
1780 .exit
```

```
 1790 PLA
 1800 .mainRoutine
 1810 JMP (oldVec)
 1820 : : : : :
 1830 .prompt1
 1840 EQUB &0D
 1850 EQUS "Press a number-key <0> to <9
>"
 1860 EQUB &0D
 1870 EQUS "specifying macro to be set u
p,"
 1880 EQUB &0D
 1890 EQUS "or press <ESCAPE> or <Q> to
quit: "
 1900 EQUB &1B
 1910 .prompt2
 1920 EQUW &0D0D
 1930 EQUS "Enter text of macro "
 1940 EQUB &1B
 1950 .prompt3
 1960 EQUD &0D0D0C07
 1970 EQUS "Macros are on."
 1980 EQUW &0D0D:EQUB &1B
 1990 .prompt4
 2000 EQUW &0D0D
 2010 EQUS "Macros are off."
 2020 EQUD &1B070D0D
```

```
 2030 .prompt5
 2040 EQUB &0D
 2050 EQUS "Macros erased and ROM disabl
ed."
 2060 EQUD &1B070D0D
 2070 .cls
 2080 EQUD &1B0D0F0C
 2090 :
 2100 .oldVec
 2110 BRK
 2120 .oldVecHi
 2130 BRK
 2140 .counter
 2150 BRK
 2160 .tallies
 2170 EQUD 0:EQUD 0:EQUW 0
 2180 : : : : :
 2190 ] NEXT
 2200 FOR N%=7 TO 4 STEP-1
 2210 IF N%?&2A1 NEXT:PRINT'"Sorry - no
free SRAM slot.":END
 2220 OSCLI "SRW. "+STR$~Z%+" "+STR$~(O%
+1)+" 8000 "+STR$ N%
 2230 N%?&2A1=&82
 2240 PRINT'"MACRO ROM running in slot "
;N%
 2250 N%=4:NEXT:END
```

Free-Text Datafiles (continued from page 18)

If we now require to read up on what John Greening has to say about fractals, we can consult the master file which will point us to a John Greening File, which we can then consult in order to find out about the articles in BEEBUG.

The important points to remember are as follows:

1) Each extract (paragraph) can exist as a separate piece of data or file or as part of a loosely assembled free-text file.

2) Movement or copying of paragraphs can be routinely performed within the word processor. Except for modifications or enhancements to extracts, the paragraph need only be entered once.

3) The method of ordering is entirely in the user's hands, and equally it can be reviewed or re-ordered as required.

4) Whatever system is used for storage of data it still requires a certain user commitment, first to collect, analyse and input the data; second, to make use of the accumulated information; and finally to ensure that others can make sense of the system.

Even with what is strictly speaking a personal system, often a second view will highlight problem areas, or make useful contribution to the methods employed. In a very real sense free-text systems are all about personal choices of what to file, where to keep the data and how it is to be used.

Wordwise User's Notebook:
The Curious Case of the Forgotten Keys

Chris Robbins puts the function keys to work in everyone's favourite word processor.

The potential word processing power of those little red keys at the top of the keyboard, and one or two others scattered around (i.e. Break, the cursor keys, and Copy) is often forgotten. But with a little imagination they can be used to provide extra functions and take a lot of tedious key pressing off your hands, or more precisely, your fingertips. There are in all 15 'soft' (programmable) keys to play with, so why not use them to take the weariness out of word processing?

For example, it's almost certain that from time to time you will want to prepare lists of some kind. They could be of anything from the extensive exotic ingredients of ancient cures for spots and other embarrassing afflictions, to the complete stores for a major expedition up the Amazon river to find the famed lost city of gold.

Whatever it is, it's equally certain that, just as with a list on paper, you will want to run a finger down your list to check things off and to mark specific items. That's easy enough on paper, but not so easy on a monitor screen, apart from which, touching the screen leaves a greasy finger smear.

HIGHLIGHTING & MARKING

True, the standard cursor is available, but it can only mark one character at a time. The following, however, overcomes all these little difficulties, and allows an 'electronic finger' to be run up and down the text on screen, and, where required, to mark items/lines for later reference or to 'tick them off', e.g. "piranha repellent packed", or perhaps "check availability of snake-proof trousers".

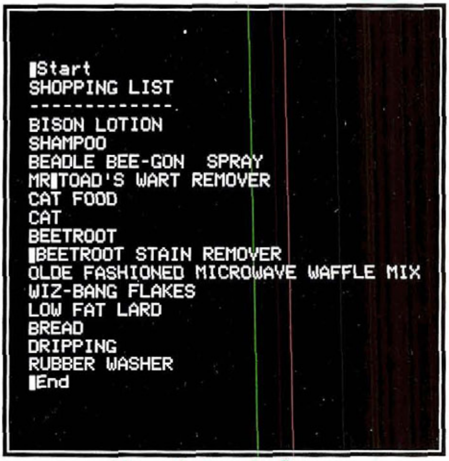
The solution is simple, and implemented by means of a set of key defined operations that apply equally to both Wordwise (*WW*) and Wordwise Plus (WW+). Thus:

```
*|WW+Bar
*|Keys f0, f1, Cursor Up, Cursor Down
*|definitions to produce a movable
*|highlighter bar
*|
*|Highlight Bar on
*KEY0"|!|L|!!"
*|Highlight Bar off
*KEY1"|!|L|A"
*|Move Bar Down (Cursor Down)
*KEY14"|!|L|A|!.|!!"
*|Move Bar up (Cursor Up)
*KEY15"|!|L|A|!/|!!"
<Return>
<Return>
```

This consists of a set of commands to define the function keys f0 and f1, and the cursor Up/Down keys. Lines beginning *| are merely comments included to explain the significance of the key definitions. It's best to set such definitions up by means of an EXEC file, as in the above where I've also included the EXEC file name WW+Bar as a comment (all comment lines will be ignored by the operating system).

The two blank lines included in the file, indicated by <Return>, avoid the necessity for extra key presses when the file is executed.

Operation is simplicity itself; having built the EXEC file using WW or WW+ it can be executed at any time from Menu mode to set up the key functions by using *EXEC WW+Bar. In Edit mode these functions are invoked in the normal way by pressing Shift-Ctrl and the appropriate key e.g. Shift-Ctrl-f0 highlights the current line of text, whilst Shift-Ctrl-f1 removes the highlight. Beware, though, that if a line isn't highlighted in the first place this latter operation will delete the first character on the line. To move the highlight bar down the screen, press Shift-Ctrl-Cursor Down, and conversely Shift-Ctrl-Cursor Up to move it up the screen.



▮Start
SHOPPING LIST

BISON LOTION
SHAMPOO
BEADLE BEE-GON SPRAY
MR▮TOAD'S WART REMOVER
CAT FOOD
CAT
BEETROOT
▮BEETROOT STAIN REMOVER
OLDE FASHIONED MICROWAVE WAFFLE MIX
WIZ-BANG FLAKES
LOW FAT LARD
BREAD
DRIPPING
RUBBER WASHER
▮End

Your electronic finger

Highlighting is achieved by placing an f1 (Green) marker at the very beginning of the line, thus turning the whole line into an embedded command; one that stays in place when the list is saved. This has the advantage that items can be marked for future use or reference, or conversely be treated as 'ticked off' since, as embedded commands, they will not appear in previewed or printed output.

A WW+ ALTERNATIVE

Although the key definition is perfectly adequate and works for both WW and WW+, a slightly more sophisticated version, albeit somewhat slower, could also be implemented in WW+ as a segment program. For instance, the following segment program provides a simple on/off highlight 'toggle' function.

```
CURSOR AT 0
C%=ASC(GCT$)
IF C%=13 THEN CURSOR UP
IF C%=2 THEN GOTO switchoff
CURSOR AT 0
FKEY1
DISPLAY
END
.switchoff
CURSOR AT 0
DELETE AT
DISPLAY
END
```

If set up in segment 0, say, repeated operation of Shift-f0 within Edit mode will switch highlighting on and off for the current line.

As with the key definition above, moving a highlighting bar is simply a matter of switching off the current line, moving to another, and switching it on again. The two short programs that follow provide just such a facility.

Program 1 - Move Highlight Bar Down

```
CURSOR AT 0
C%=ASC(GCT$)
IF C%=13 THEN CURSOR UP
IF C%=2 THEN DELETE LEFT
CURSOR AT 0
CURSOR DOWN
FKEY1
DISPLAY
END
```

Program 2 - Move Highlight Bar Up

```
CURSOR AT 0
C%=ASC(GCT$)
IF C%=13 THEN CURSOR UP
IF C%=2 THEN DELETE LEFT
CURSOR AT 0
CURSOR UP
FKEY1
DISPLAY
END
```

Program 1 could be placed in segment 1, and program 2 in segment 2, then in Edit mode, pressing Shift-f1 will move a highlight bar down, and Shift-f2 move it up. Using WW+ segment programs in this way is admittedly slower, but it's a safer approach, since the presence of an f1 (Green) marker at the beginning of a line can be checked, thus avoiding inadvertent deletion of characters. It isn't foolproof though, and the programs can be misled by embedded commands at the beginning of lines.

It's also possible to use the cursor Up/Down keys in a similar way to the key definition approach given earlier. For instance, by defining keys 14 and 15 as follows:

```
*|Move Bar Down (Cursor Down)
*KEY14|[:DOLINE"||GSEG1"|M|[|[
*|Move Bar Up (Cursor UP
*KEY15|[:DOLINE"||GSEG2"|M|[|[
```

The programs in segments 1 and 2 can be invoked by pressing Shift-Ctrl-Cursor Up and Shift-Ctrl-Cursor Down to move the highlight bar up and down, but it's a rather longwinded and slow way of doing things. Better by far to move the highlighter by Shift-f0 and Shift-f2. However, it does provide a good illustration of a mechanism whereby segment programs can be called up from programmable keys other than the red function keys.

Incidentally, on-screen highlighting in the way I've described is also a real boon when proof reading text.

SEGMENT JUMPING

WW+ users have an advantage over WW users in that they have an additional ten independent areas (segments) of memory available to them, and which can be used for developing and running segment programs, or as often as not simply as scratchpad work areas. For example, it's possible to have 11 entirely separate pieces of text, documents, sets of notes, letters, or whatever, in memory simultaneously, and to swap between them at will.

However, going from main text to another segment (for instance), means leaving Edit mode, selecting option 9 to call up the Segment menu, selecting option 5 from this menu, then the required segment, followed by Escape to enter Edit mode again; a procedure that requires up to five key presses. A much simpler approach is to make use of the programmable keys to go directly from text area to text area in Edit mode. For example, function keys 0 to 9 might be programmed to select the equivalent segments, whilst function key 11, the Copy key, could return control to the main text area, thus:

```
*KEY0"|[:SE.SEG.0|M|M|["
*KEY1"|[:SE.SEG.1|M|M|["
*KEY2"|[:SE.SEG.2|M|M|["
*KEY3"|[:SE.SEG.3|M|M|["
*KEY4"|[:SE.SEG.4|M|M|["
*KEY5"|[:SE.SEG.5|M|M|["
*KEY6"|[:SE.SEG.6|M|M|["
*KEY7"|[:SE.SEG.7|M|M|["
*KEY8"|[:SE.SEG.8|M|M|["
*KEY9"|[:SE.SEG.9|M|M|["
*KEY11"|[:SE.TE.|M|M|["
```

Having set up these key definitions (preferably by means of an EXEC file), pressing Shift-Ctrl and the appropriate key in Edit mode allows you to jump about at will; between main text area and any segment, segment to main text area, and directly from segment to segment. And all this without affecting the normal key operations in any way; segment programs can still be run (assuming they're present) by Shift-fn(segment number), and the operations usually associated with the function keys and others will all work as normal.

This does of course mean that most of the programmable keys (Shift-Ctrl functions) have been used up. But the cursor keys, and of course Break, are still available to play around with. If they aren't enough, an alternative approach would be to use a small program, in segment 0 say, to provide the selection and switching mechanism.

SAVING FILES

Yet another tedious, and on occasion 'finger problem' prone business, is that of saving a text file. In WW this would involve pressing Escape to leave Edit mode, selecting option 1 from the menu, typing in the file name, pressing RETURN, and, subsequent to the save, pressing Escape to return once again to Edit mode. The additional overwrite check in WW+ requires yet another key depression. Late at night when the candle's burning low, and after many hours of editing and innumerable saves of the same file, this could begin to prove just a mite irksome; the fingers could stray or slip, and disaster strike. So why not let the keys take the strain? For example in WW, the key definition to make the Copy key do all the work might be

```
*KEY11"|[1:3.W.FRED|M|["
```

where :3.W.FRED is an example of a file name. In WW+ this definition would be extended to include "Y" to the "Replace old file" question i.e.

```
*KEY11"|[1:3.W.FRED|MY|["
```

Having defined the Copy key in this way, subsequent depression of Shift-Ctrl-Copy from Edit mode will save the file, and return you from whence you came in Edit mode.

A KEYED UP MENU

In similar fashion, if other menu options are to be used often and without any change, it makes sense to program keys to provide direct operation from within Edit mode. In fact operation of the whole menu (Main or Segment menu in WW+) could be assigned to programmable keys, e.g. f1 - Save entire text, f2 - Load new text, f3 - Save marked text etc. For the sake of completeness here's the full Main menu list for WW+ (WW users omit the KEY9 definition and the Y in KEY1 and KEY3). The triangular brackets aren't part of the key definitions, they merely serve to delimit variable items such as file names, which you should alter yourself.

```
*KEY1"|[1<filename>|MY|["
*KEY2"|[2<filename>|M|["
*KEY3"|[3<filename>|MY|["
*KEY4"|[4<filename>|M|["
*KEY5"|[5<G or S><search
string>|M<replace string>|M|[
*KEY6"|[6|M|["
*KEY7"|[7|M|["
*KEY8"|[8<filename>|M|["
*KEY9"|[95<segment number>|["
```

This may seem a little extravagant in the use of programmable keys, but it rather depends on how much use is to be made of the various operations invoked via the menu, and your powers of concentration in the wee small hours.

An optional extension to the KEY3 definition (*Save marked text*) would be to delete the markers in text, following the save. The complete KEY3 definition would then be:

```
*KEY3"|[3<filename>|MY|[@|!|O|!$|
!#|A|!$|!#|A|!|O|!$@|A"
```

Put in slightly more intelligible form this is equivalent to the following sequence of operations:

1) Escape (to Main menu)
2) Select Main menu option 3
3) Type file name
4) Return
5) Reply "Y" to *Replace old file* (WW+ only)
6) Escape (to Edit mode)

continued on page58

Graphics

Alan Wrigley starts a new series of 1st Course articles on graphics programming

It has been quite a while since 1st Course tackled the subject of graphics, so we decided it was time to look at this topic again. In its day, the BBC micro's graphics capability was legendary, and it was possible to do things that most other micros around at the time just could not achieve. It took the arrival of 16- and 32-bit micros to push the standards beyond those which had been set by the humble BBC.

What this means in effect is that, even today, the BBC is still capable of producing graphics of outstanding quality, and it is therefore well worthwhile getting to grips with the programming techniques involved. This is made all the easier by the existence of BBC Basic, which was designed with graphics commands as an integral part of the language, and not as an afterthought as with many other Basic dialects.

SCREEN MODES

Before we get down to programming in detail, we need to take a brief look at the way in which the machine is structured with regard to the various screen modes, since this affects the way graphics are handled. Mode 7 (and its equivalent shadow mode on the Master, mode 135) is a teletext mode and operates in a completely different way from all the other modes. In this mode, it is only possible to address the screen by one whole character position at a time. Whereas with the other modes text characters are defined pixel-by-pixel and plotted on the screen as such, in teletext mode the whole character is defined in the teletext chip itself and plotted on the screen as a discrete entity. Graphics *are* possible, but only by using special graphics characters provided in the character set; these are obviously limited since the ASCII set only has 255 characters to start with and half of these are taken up with control codes and the normal character set.

As a result of all this, graphics in mode 7 have a very chunky appearance. A previous *1st Course* series (Vol.7 No.10 - Vol.8 No.4) provided a detailed look at using text and graphics in teletext modes, and we will not consider them further in the current series.

NON-TELETEXT MODES

All other modes in the BBC micro are pixel-based. However, modes 3 and 6 and their shadow equivalents are text-only modes, so although the characters are pixel-based (and can be redefined using VDU23), it is not possible to produce graphics in these modes. Issuing a graphics command in modes 3 or 6 simply has no effect at all.

This leaves modes 0,1,2,4 and 5 (and their shadow equivalents). In these modes, each individual pixel on the screen can be separately addressed, allowing you to draw graphics of any shape and size anywhere on the screen. One of the features that made the BBC so powerful alongside its original contemporaries is that text and graphics can be mixed freely wherever you require them. Many other micros of the early eighties had text screens for text and graphics screens for graphics; the only way of getting text onto a graphics screen was to draw the characters yourself.

CO-ORDINATES

To make life even easier for the programmer, BBC Basic operates two entirely separate sets of screen co-ordinates. Text co-ordinates follow the principle of character positions, which we referred to earlier when we discussed mode 7. For example, in mode 0 which has 80 columns by 32 rows, the normal PRINT command can place text at any one of 80 horizontal and 32 vertical positions (with 0,0 being at the top left corner of the screen). The actual characters themselves are still plotted as pixels, but the operating system takes care of the positioning so that as far as the programmer is concerned, he can print text on an 80x32 grid without even being aware of the underlying pixel co-ordinates.

The graphics co-ordinates, on the other hand, allow individual pixels to be specified. This enables lines to be drawn, shapes to be constructed and even single pixels to be plotted, all precisely positioned wherever you want them. All this can be freely mixed on the same screen with text printed using text co-ordinates. And as if this were not powerful enough in itself, it is also possible to print text at the graphics co-ordinates, meaning that your text does not have to be constrained by the text grid if you don't want it to be.

BRASS TACKS

This is starting to sound like an awful lot of theory, whereas in practice it is actually relatively simple. So without further ado we will get down to brass tacks and see how to put the theory into practice. For the moment we will assume that we are working with the full screen and not restricting our output to a particular area by setting up a window. First of all, how do we know which set of co-ordinates we are using? This is quite simple - if we issue a graphics command (DRAW, MOVE, PLOT or their VDU25 equivalents), then the parameters to the command will be taken as graphics co-ordinates. If on the other hand we send a printable character to the screen, it will be printed using the text co-ordinates *unless* we have previously told the computer that all text output should be printed at the graphics co-ordinates.

Unlike the text co-ordinates, graphics co-ordinates start with 0,0 at the *bottom* left corner of the screen, and thus increase horizontally from left to right and vertically from bottom to top, just like Cartesian co-ordinates as used in graphs. *All* graphics co-ordinates, irrespective of screen mode, are given in the range 0-1279 horizontally and 0-1023 vertically. These are often referred to as *OS co-ordinates* (indicating that the co-ordinates are maintained by the operating system independently of the screen mode and hence also of the number of pixels on the screen). In other words, when you are working with graphics, the screen is *always* divided into 1280 units horizontally and 1024 vertically, and you should get used to working with these values regardless of which mode you are using.

Let us try putting this into practice. Type in the following lines and run the program:

```
10 MODE 0
20 MOVE 0,0
30 DRAW 1279,1023
```

You should see a diagonal line running from the bottom left of the screen to the top right. Now alter the mode number at line 10 to any of the other graphics modes (1,2,4 or 5) and run it again. Although we have not altered the specified co-ordinates, the line appears in exactly the same place irrespective of the mode, though you will notice that it varies in thickness depending on the resolution of the mode, and hence the width of the individual pixels in that mode.

This illustrates an important point that you must sometimes bear in mind. Whereas for many purposes you do not need to consider the resolution, since the system of OS co-ordinates ensures that the image is always at the same position on the screen irrespective of mode, there *are* instances where taking the resolution into account can improve your programs. Take animation, for instance; if you are moving an object across the screen in, say, mode 2, there is absolutely no point in trying to get smoother animation by moving it by one or two OS co-ordinates at a time, since the minimum distance an object can move in any mode is one pixel. In mode 2 this is equivalent to 8 OS units.

The purpose of the graphics commands we have used here, MOVE and DRAW, should be fairly self-explanatory from their names, and we will be looking at some uses for these commands in next month's article. We will also be looking at the question of colour later in the series, since we have not yet mentioned this and it is of course an important element in the use of graphics. To finish off this month, though, we will return to the facility to print text at graphics co-ordinates rather than text co-ordinates. To do this we make use of the command VDU5.

VDU5

This command tells the operating system that from this point in the program onwards we want all text output, i.e. all printable characters sent to the screen, to be printed at the current graphics cursor. Having issued a VDU5, you can then print anywhere on the screen, usually after using a MOVE statement to move the graphics cursor to the required position. This does not have to coincide with a normal text character position. For example, type in the following short program and run it:

```
10 MODE 0
20 VDU5
30 FOR i%=0 TO 25
40 MOVE i%*24,1023-i%*4
50 PRINT CHR$(i%+65)
60 NEXT
70 VDU4
```

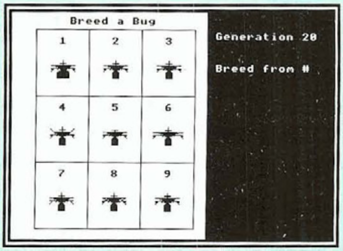
You should see the letters A to Z printed along a line which slopes gently down the screen. What we have done here is to set VDU5 in line 20, and then entered a loop at line 30. Line 40 moves the graphics cursor each time round the loop, starting at 0 in the x direction and adding 24 OS units each time, and at 1023 in the y direction but this time subtracting 4 units each time. Line 50 prints the next character in the alphabet at the new co-ordinates - because we are starting at ASCII 65 ("A"), we add 65 to the value of the loop variable.

You can try experimenting with this facility. With a little practice you can print text in circles or undulating waves (though the characters themselves will always be printed at the same orientation, i.e. upright).

To cancel VDU5 and return to normal text co-ordinate printing you use VDU4. The latter is the default state when the computer is switched on or reset, and is also set after a mode change. If you write a program which uses VDU5 you should make sure that VDU4 is *always* reset when the program exits, since otherwise the normal text cursor will not be visible and the user may well get confused.

Best of BEEBUG

Applications I Disc



BUSINESS GRAPHICS - for producing graphs, charts and diagrams
VIDEO CATALOGUER - catalogue and print labels for your video cassettes
PHONE BOOK - an on-screen telephone book which can be easily edited and updated
PERSONALISED LETTER-HEADINGS - design a stylish logo for your letter heads
APPOINTMENTS DIARY - a computerised appointments diary
MAPPING THE BRITISH ISLES - draw a map of the British Isles at any size
SELECTIVE BREEDING - a superb graphical display of selective breeding of insects
THE EARTH FROM SPACE - draw a picture of the Earth as seen from any point in space
PERSONALISED ADDRESS BOOK - on-screen address and phone book
PAGE DESIGNER - a page-making package for Epson compatible printers
WORLD BY NIGHT AND DAY - a display of the world showing night and day for any time and date of the year

File Handling for All

on the BBC Micro and Acorn Archimedes

by David Spencer and Mike Williams

File Handling for All
on the BBC Micro and Acorn Archimedes
David Spencer and Mike Williams

Computers are often used for file handling applications yet this is a subject which computer users find difficult when it comes to developing their own programs. *File Handling for All* aims to change that by providing an extensive and comprehensive introduction to the writing of file handling programs with particular reference to Basic.

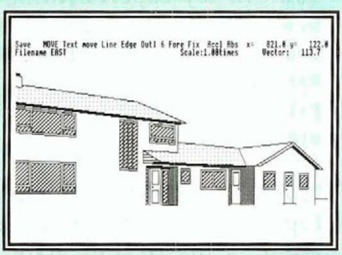
File Handling for All, written by highly experienced authors and programmers David Spencer and Mike Williams, offers 144 pages of text supported by many useful program listings. It is aimed at Basic programmers, beginners and advanced users, and anybody interested in File Handling and Databases on the Beeb and the Arc. However, all the file handling concepts discussed are relevant to most computer systems, making this a suitable introduction to file handling for all.

The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in-depth look at the handling of different types of files e.g. serial files, indexed files, direct access files, and searching and sorting. A separate chapter is devoted to hierarchical and relational database design, and the book concludes with a chapter of practical advice on how best to develop file handling programs.

The topics covered by the book include:

Card Index Files, Serial Files, File Headers, Disc and Record Buffering, Using Pointers,
Indexing Files, Searching Techniques, Hashing Functions, Sorting Methods,
Testing and Debugging, Networking Conflicts, File System Calls

The associated disc contains complete working programs based on the routines described in the book and a copy of Filer, a full-feature Database program originally published in BEEBUG magazine.


Save MOVE Text move Line Edge Outl 6 Fore Fix Accl Abs x= 821.8 y= 122.8
Filename EAST Scale:1.00times Vector: 113.7

ASTAAD

Enhanced ASTAAD CAD program for the Master, offering the following features:

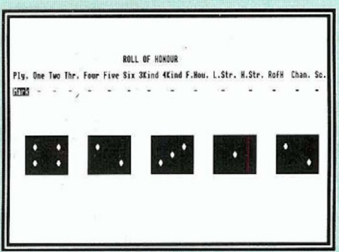
- full mouse and joystick control
- built-in printer dump
- speed improvement
- STEAMS image manipulator
- Keystrips for ASTAAD and STEAMS
- Comprehensive user guide
- Sample picture files

	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407a	£5.95	**ASTAAD** (3.5" ADFS)	1408a	£5.95
Applications II (80 track DFS)	1411a	£4.00	**Applications II** (3.5" ADFS)	1412a	£4.00
Applications I Disc (40/80T DFS)	1404a	£4.00	**Applications I Disc** (3.5" ADFS)	1409a	£4.00
General Utilities Disc (40/80T DFS)	1405a	£4.00	**General Utilities Disc** (3.5" ADFS)	1413a	£4.00
Arcade Games (40/80 track DFS)	PAG1a	£5.95	**Arcade Games** (3.5" ADFS)	PAG2a	£5.95
Board Games (40/80 track DFS)	PBG1a	£5.95	**Board Games** (3.5" ADFS)	PBG2a	£5.95

All prices include VAT where appropriate. For p&p see Membership page.

RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS.

Board Games


ROLL OF HONOUR
Ply. One Two Thr. Four Five Six 3Kind 4Kind F.Hou. L.Str. H.Str. RofH Chan. Sc.

Solitaire - an elegant implementation of this ancient and fascinating one-player game, and a complete solution for those who are unable to find it for themselves.

Roll of Honour - Score as many points as possible by throwing the five dice in this on-screen version of 'Yahtze'.

Patience - a very addictive version of one of the oldest and most popular games of Patience.

Elevenses - another popular version of Patience - lay down cards on the table in three by three grid and start turning them over until they add up to eleven.

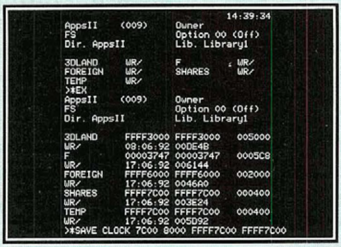
Cribbage - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.

Twiddle - a close relative of Sam Lloyd's sliding block puzzle and Rubik's cube, where you have to move numbers round a grid to match a pattern.

Chinese Chequers - a traditional board game for two players, where the object is to move your counters, following a pattern, and occupy the opponent's field.

Aces High - another addictive game of Patience, where the object is to remove the cards from the table and finish with the aces at the head of each column.

Applications II Disc



```
                                      14:39:34
AppsII      (009)       Owner
FS                      Option 00 (Off)
Dir. AppsII             Lib. Library1

3DLAND      WR/         F            WR/
FOREIGN     WR/         SHARES       WR/
TEMP        WR/
>*EX
AppsII      (009)       Owner
FS                      Option 00 (Off)
Dir. AppsII             Lib. Library1

3DLAND      FFFF3000 FFFF3000    005000
WR/         08:06:92 00DE4B
F           00003747 00003747    0005C8
WR/         17:06:92 006144
FOREIGN     FFFF6000 FFFF6000    002000
WR/         17:06:92 0046A0
SHARES      FFFF7C00 FFFF7C00    000400
WR/         17:06:92 003E24
TEMP        FFFF7C00 FFFF7C00    000400
WR/         17:06:92 005D92
>*SAVE CLOCK 7C00 8000 FFFF7C00 FFFF7C00
```

Crossword Editor - for designing, editing and solving crosswords

Monthly Desk Diary - a month-to-view calendar which can also be printed

3D Landscapes - generates three dimensional landscapes

Real Time Clock - a real time digital alarm clock displayed on the screen

Running Four Temperatures - calibrates and plots up to four temperatures

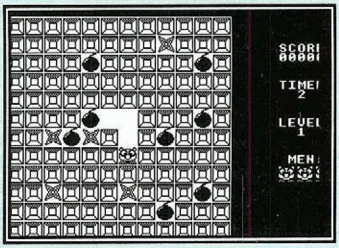
Julia Sets - fascinating extensions of the Mandelbrot set

Foreign Language Tester - foreign character definer and language tester

Share Investor - assists decision making when buying and selling shares

Label Processor - for designing and printing labels on Epson compatible printers

Arcade Games



George and the Dragon - Rescue 'Hideous Hilda' from the flames of the dragon, but beware the flying arrows and the moving holes on the floor.

Ebony Castle - You, the leader of a secret band, have been captured and thrown in the dungeons of the infamous Ebony Castle. Can you escape back to the countryside, fighting off the deadly spiders on the way and collecting the keys necessary to unlock the coloured doors?

Knight Quest - You are a Knight on a quest to find the lost crown, hidden deep in the ruins of a weird castle inhabited by dangerous monsters and protected by a greedy guardian.

Pitfall Pete - Collect all the diamonds on the screen, but try not to trap yourself when you dislodge the many boulders on your way.

Builder Bob - Bob is trapped on the bottom of a building that's being demolished. Can you help him build his way out?

Minefield - Find your way through this grid and try to defuse the mines before they explode, but beware the monsters which increasingly hinder your progress.

Manic Mechanic - Try to collect all the spanners and reach the broken-down generator, before the factory freezes up.

Quad - You will have hours of entertainment trying to get all these different shapes to fit.

	Stock Code	Price
File Handling for All Book	BK02b	£9.95
File Handling for All Disc (40/80T DFS)	BK05a	£4.75
Joint Offer book and disc (40/80T DFS)	BK04b	**£11.95**
Magscan (40 DFS)	0005a	£9.95
Magscan (80T DFS)	0006a	£9.95
Magscan (3.5" ADFS)	0007a	£9.95

	Stock Code	Price
File Handling for All Disc (3.5" ADFS)	BK07a	£4.75
Joint Offer book and disc (3.5" ADFS)	BK06b	**£11.95**
Magscan Upgrade (40 DFS)	0011a	£4.75
Magscan Upgrade (80T DFS)	0010a	£4.75
Magscan Upgrade (3.5" ADFS)	1458a	£4.75

All prices include VAT where appropriate. For p&p see Membership page.

512 Forum

by Robin Burton

As promised at the end of last month's 512 Forum we are having a break this time from CHKDSK.

I've had dozens of requests for information on application compatibility over the last four years, and now there's a chance to do something about them. While I do, there's another closely related matter that I ought to deal with at the same time.

COMPATIBILITY ISSUES

Most 512 Forum readers are well aware of my involvement in the company Essential Software (if you weren't, you are now) but they'll also know that I rarely mention it in the Forum. Mike Williams and I have always thought it best to avoid the possibility of suggestions that I'm abusing my position, but it must be said that sometimes there's a detrimental effect on the contents of the Forum.

In fact I've had what amount to complaints from quite a number of readers who are also Essential Software customers. Their gripe is that it's unreasonable for me to have to avoid obvious or important 512 topics that are of interest to them for any reason. The argument goes: "BEEBUG is the only magazine that has ever provided real 512 support, while Essential Software is unique in its range of 512 products. Since I'm the link between the two, why shouldn't I take advantage of the situation if it can benefit readers and customers alike?"

Most of you probably appreciate this is a classic *Catch 22* situation. Information I come by with my Essential Software hat on, because of or about its products, isn't used in the Forum precisely because that's its source. We're now looking at software compatibility, which is a perfect example where too much modesty is particularly counter-productive for Forum readers.

The basis of the problem is, as usual, Acorn's virtually non-existent support for the 512. That amounted to a single operating system upgrade since the system was launched. Even so, not only was the original bundled operating system several years out of date when the machine was launched, but so was the upgrade version, which was also dated to before the machine was launched. The only possible reason I can think of for this was that since these were out of date versions of DOS they were extremely cheap to licence, allowing Acorn to maximise profits.

If you think that's an overly harsh judgement, I must disagree. As justification, here are the facts. The 512 first appeared in 1986, but throughout its life as a current product it was only ever supplied with DOS Plus 1.2. Version 2.1 was always an upgrade which the user had to acquire separately, and for several years this was at an extra cost of £15.00. By contrast, MS-DOS and PC-DOS had reached version 2.1 as early as 1983, while MS-DOS 3.1 was released before the end of 1984. This was still over a year before the 512's arrival. By the time the 512 arrived both MS-DOS and PC-DOS had reached version 3.2 with 3.3 soon to follow.

As a result of Acorn's policies, if software compatibility is a problem there are very few options open to the 512 user. For example, you can't do much about hardware problems caused by the BBC micro host system, nor can you update the operating system to a later version than DOS Plus 2.1. In fact the only effective steps you can take with software involve adding third party programs to make up for some of the issue software's deficiencies or omissions, while hardware options are limited to expanding the memory of your 512. My problem in mentioning either of these is that there's only one supplier offering such upgrades.

Some of the compatibility information users have told me about over the last couple of years, for example, is solely as a result of them adding a memory expansion, but I've avoided mentioning it too explicitly until now. I think it's time that changed. Obviously, virtually all programs benefit from more free RAM, but the difference in the 512's case is more than just being able to handle bigger files or a performance improvement in a user's existing software.

I won't labour the point any more, but expanding your 512's memory is just about the only way genuinely to improve its overall software compatibilty.

GENERAL COMPATIBILITY

There are quite a number of particularly well known PC programs that won't run in a standard memory 512, but which do run perfectly happily in an expanded machine. The thing is that when many of these programs are run in a 512K machine they crash with the the all too frequent 'Invalid Opcode' error, or less common crashes such as 'Overflow' or occasionally 'single step', which normal programs should never generate.

Some programs crash the entire system, while others don't crash but will refuse to run, reporting such spurious problems as 'Invalid version of DOS', 'Out of memory at <address>', or even worse 'Needs DOS version 3'. Of course a few are better behaved and more helpful, simply informing you that they need more RAM. Those are the easy ones.

On the face of it a good many of these programs seem to be a lost cause in the 512, given the sort of problem they appear to exhibit if you try them in a standard memory machine. Even if you know some of these errors aren't accurate, given the wide range of possible results, there's little one can deduce about whether a particular program has a genuine, permanent problem or simply needs more RAM.

However, many of these apparent problems are totally misleading and in some cases an increase in the 512's memory is all that's needed to cure them. Amongst the programs that I can call to mind, most of the following either won't run at all in a standard 512, variously crashing the system entirely or giving misleading errors, or they have serious restrictions or problems that render them effectively unusable. All I can say is thanks to those users who were stubborn enough to try again with a program they knew from past experience would fail. These didn't.

WordStar v5.1 will work in a standard machine (I think) but memory is desperately tight and the spelling checker and thesaurus are unusable. In an expanded machine the application is fine, including the ability to store the dictionary or the thesaurus in RAM ready for instant access.

WordPerfect 5 reportedly either won't install or crashes in a standard machine (I can't remember which it is), but like WordStar above, it's fine in an expanded 512.

GEM 3 and Timeworks I've mentioned before. The packages must be installed on a real PC and moved complete to the 512, but they're then perfectly OK. Without a memory expansion GEM 3 is simply much too large to run anything worthwhile.

MS-Word 5 is perfectly happy in an expanded 512, but not in a standard machine. Either it can't be loaded at all, or it leaves too little RAM to do anything useful. Again I can't remember which, but users with memory expansions are completely happy with it.

SuperCalc 5 (a version bought in America) seemingly works acceptably well in an expanded 512 (apart from obvious key and character differences) but it too won't run in a standard machine.

One user informed me that the Smart System (word processor, spreadsheet, database, notepad, etc.) seems to work well in his expanded 512. Mind you, the chances are that if you can afford the cost of this package you probably also have a PC too.

Ventura Publisher is another package that won't run in a standard machine, but a memory expansion fixes the problem. However, this is another package costing several hundred pounds, so, like Smart, it's not likely to interest many domestic users.

COMPATIBLE SOFTWARE

Many thanks to the two Forum readers who've responded to my appeal for application compatibility information. No doubt many more of you will be writing to me in due course with even more information (he said more in hope than expectation). The first three offerings are my own contribution, but the rest were provided by just two readers, so come on the rest of you, share your knowledge and help other 512 users.

All the programs below run in a standard memory machine using DOS Plus 2.1, unless otherwise stated. Comments in quotes are reproduced from readers' letters exactly as received. In shareware programs you'll find full details of costs and how to register in the documentaion file on the disc, though indications are given below.

PC-Write, previously marketed in the U.K. by Sagesoft, is now only available as shareware. It's written by Quicksoft of Seattle, Washington. I've used version 2.7, 3.03 and 3.04. All work perfectly, even with a model B host, so long as a few key functions are re-assigned, which you can do permanently or temporarily in the package. Earlier versions still exist and might work too, but I can't guarantee them. Note that from version 3 onwards the main program, ED.EXE, is over 225K, so memory is extremely tight in a standard 512, preventing use of the spelling checker and/or of the on-line help if a document is loaded. Registration costs $129.00, though there are options within this.

A86/D86 is without a doubt THE assembler debugger package for '86 series machines, both for PCs in general, but especially the 512 in particular. It's available as shareware and was written by Eric Isaacson who supports the package directly. Amazingly, absolutely everything works in the 512 except breaking out of a free running tight code loop in the debugger (because break is non-functional in the 512). The latest version is 3.02. Full registration is $97.00 including postage to the U.K., or via Shareware Marketing. Again there are registration options.

PC-Tools de Luxe by Central point Software is probably the most well known commercial file management package. I have version 5.5 and almost everything works on PC formatted discs (including the 512's winchester) except those functions relying on direct hardware access to the disc. Unsurprisingly it does not like the 512's 800K or 640K floppies, reporting 'FAT corrupt'. I haven't tried a later version than 5.5, though it's now up to about v7.1. Cost varies. If you shop around you can get it for about £60-£70.00. Program modules are large and the installed system needs well over 1.5Mb of disc, so it's restricted to winchester users and probably needs an expanded memory machine too.

Q-Edit by SEMware is a shareware text editor and version 2.15 "works perfectly". A second copy of COMMAND.COM must be loaded to install or to run it, or use Essential Software's new FIXEXE program.

As-Easy-As from Trius is also shareware, but is a spreadsheet very similar to (some say better than) Lotus 1-2-3. Version 4.00 works very well. Version 5.01 also works, but not quite so well. There are one or two screen quirks and clearly it isn't so well suited to a mono display. Version 4 is very memory restricted and version 5 probably wouldn't run at all in a standard memory 512. Again, a second copy of COMMAND.COM or FIXEXE is also required.

PC-Lite v1.03 "seems to work well". My correspondent suggests it's probably cheaper to register in the U.S. than via a U.K. shareware outlet. Registration is $79.00 + $30.00 p&p.

PC-Browse v1.01 is a "multiple file searcher and much else, works fine but only in stand-alone mode. Use ^L and ^O for searching keys". Like many programs, it won't respond to keys if run as a TSR. Registration is $79.00 + $30.00 p&p.

Treeview (no version number supplied) is a shareware disc manager which "works fine and looks useful as a DOS shell for copying, backups, etc". Launching applications from it "leaves too little memory in a standard system to do much". Registration is £25.00 via U.K. shareware outlets.

InfoSelect v2.0 by Micrologic is a "paste-it type note taking package". It too can't be used in TSR mode, but is fine if run as a foreground program. It's available through shareware outlets, or from Compuclassics, 7959 Deering Avenue, Canoga Park, CA 91304, U.S.A. Registration is $89.00 + $30.00 p&p.

Lotus Magellan v2.0 is "available anywhere at about £85.00, looks as if it may work well as a stand-alone but needs a hard disc to evaluate properly". That's all I have on this; I don't even know what it does.

PC-Style v1.0 "works fine". Shareware. No other details given.

Getit v1.0 is a shareware "simple free-form database. Works fine except for page up and page down". Registration unknown.

INCOMPATIBLE SOFTWARE

Often just as useful as knowing what will run in the 512 is knowing which programs won't, so here's a list of programs that belong to this category. Again they're a mixture of my own information and that supplied by the two members.

Laplink v1.35/2.16, Travelling Software (U.S.) Like virtually all PC comms programs the problem is with the serial port.

Laplink Pro v4.0a would be afflicted as above, but it won't even load.

NOTE: It's really not worth the effort of even trying PC comms. software. Only Kermit (PD) and COMM+ are known to work, and even here COMM+ is a special 512 version.

PC-Write Standard Level v2, (Travelling Software) fails with an invalid opcode.

PC-Outline v3.34 from Brown Bag Software (shareware) doesn't load, although v1.08 supposedly does work.

These last two were tried in a standard memory system, so they might work in an expanded 512.

Galaxy Lite v1.70 from Starlite Software doesn't work, even in an expanded 512. It can be installed and loaded, but it then freezes the machine.

That's all the compatibility information I have available immediately, so if you'd like to add to it, write to me c/o BEEBUG. In the meantime, it's back to CHKDSK in next month's issue.

High Security Ciphers (Part 2)

Ruben Hadekel concludes his investigation into codes by turning theory into practice.

Last month I explained how the BBC series of computers could be used to create complex and very secure encryption of a text message. This concluding article extends the process and shows ways of producing even more obscure keys.

Hush, the third version of the program, is listed at the end of this article (see listing 2). Just type in the program and run it. This version is a departure from classical cryptographic practice in that the message can contain both lower and upper case characters. In fact, any keyboard character except '{' and '}', can be coded and decoded. Spaces and carriage returns are also reproduced so that not only the subject matter of the text is enciphered, but also its format.

To make the best use of this feature, the program as listed begins with MODE 131, which works with Master series micros only (and BBC B+). With a BBC B alter line 100 to MODE 3. Unless the micro is fitted with shadow RAM, the amount of text that can be displayed in mode 3 will be limited.

If entered directly via the keyboard, the message or the encrypted text must be terminated by the '~' character instead of '|', the latter being an allowable text character. The '{' and '}' characters are used by the program to replace spaces and carriage returns respectively before enciphering, and if used in error in the text they are simply ignored. They may well, however, appear in the encrypted message, and must then be typed in for decoding.

EVEN SAFER

If a character is enciphered by a random number without restriction, then it can be enciphered by itself on occasion. With the Sphinx or Sesame programs, the odds of this happening are 1 in 26 per character.

Let us assume that a sequence of letters enciphered by themselves can be of interest if the number of letters exceeds three. A human analyst glancing at the enciphered text could spot such a sequence (which could of course be a chance encipherment, but that is unlikely), and this could be a useful clue to the subject matter of the message, even though it could hardly help in breaking the cipher by discovering the key.

For the Sphinx and Sesame versions, the odds against a sequence of four letters being enciphered by itself are 1-(1/26)^4. In a message of 4000 characters, the odds of there being at least one sequence of four letters enciphered by themselves are 1-(1-(1/26)^4)^4000, i.e. roughly 4000*(1/26^4), or approximately one in a hundred, and that might be considered unacceptably high.

With the Hush version, the odds of a character being enciphered by itself are 1 in 93, but for an alphabetic character the odds of it being enciphered by itself or its upper or lower case counterpart are 2 in 93. Again, let us assume that a sequence of letters enciphered by themselves can be of interest if the number of letters exceeds three. In this case the same analysis as above shows that the probability of there being at least one sequence of four or more unenciphered letters in a message 4000 characters long is somewhat less than 1 in 1170, and even that might again be considered unacceptably high.

Line 350 in all the programs ensures that no character can be enciphered by itself, or for the Hush version that no alphabetic character can be enciphered by itself or its upper or lower case counterpart.

The above analysis is simplistic, and pessimistic. With the odds against the first four characters being enciphered by themselves as stated, the odds against the next four characters being similarly enciphered are somewhat higher. An exact analysis would be quite complicated, and unwarranted. It is enough to say that the actual odds are somewhat better than the figures quoted.

KEY GENERATION

The problem of generating keys warrants discussion. Since keys may need to be changed at times, plucking a number out of the air could be dangerous, since there could be some unconscious bias, leading perhaps to repetition of a former key, a preference for certain digits etc. Using the computer's random function to generate keys could also be questioned. A truly random key can be generated by repeatedly rolling a die up to 12 times, or a set say of four dice three times. The resulting sequence of digits must be less than 354232356455. At this stage enter the short program given in listing 1, *KEY*, and save it to disc. Now run this program and enter the result of your dice throws on the prompt "Enter number". The program will then generate and display a corresponding number up to one billion minus one. Roll one die again, and prefix the number generated by KEY by one if an odd number comes up. You then have a key from zero to two billion minus one.

Keys b% and d% can be similarly generated by rolling a die six times. The resulting number must not exceed 523551 if the key is not to exceed 33000 (2134363 for 50000), but higher numbers are perfectly in order - they will merely result in increased execution times.

The program works by at first considering the sequence of digits as a number in base seven, but without any possible zeros. It then subtracts one from each digit, leaving a number written in base six, without any restrictions. That number is then converted to denary.

While these articles have been of a fairly deep and technical nature, they do demonstrate the power of even a small computer in serious and practical applications. They also show that the computer can push mathematical theories into the realm of the real world.

Listing 1

```
   10 REM Program Hush
   20 REM Version B 1.0
   30 REM Author  Ruben Hadekel
   40 REM BEEBUG  Aug/Sep 1992
   50 REM Program subject to copyright
   60 :
  100 MODE 131
  110 DIMA$(2):sq%=0
  120 ON ERROR GOTO 190
  130 J%=OPENIN("savekey")
  140 INPUT#J%,a%,b%,c%,d%:ON ERROR OFF:
CLOSE#0
  150 CLS:PRINT"Stored keys are":PRINTa%
'b%'c%'d%;'"Do these stand (Y/N)?"
  160 IF FNYN="N" THEN 190
  170 *LOAD seq 5BCC
  180 GOTO 400
  190 ON ERROR OFF:J%=OPENOUT("savekey")
  200 CLS:INPUT"Enter keys a,b,c,d "a%,b
%,c%,d%
  210 CLS:PRINT"Do you want to save the
keys to disc? (Y/N)":IF FNYN="N" sq%=1:
GOTO 230 ELSE PTR#J%=0
  220 PRINT#J%,a%,b%,c%,d%
  230 CLOSE#J%
  240 CLS:PRINT"Wait for next prompt"
  250 U%=&5BCA
  260 X%=RND(-c%)
  270 FOR X%=1 TO d%:Y%=RND(1000):NEXT
  280 FOR X%=1 TO 4000:Y%=RND(1000):U%=U
%+2
  290 ?U%=Y% DIV 256:?(U%+1)=Y% MOD 256
  300 NEXT X%
  310 U%=&5BCA
  320 X%=RND(-a%)
  330 FOR X%=1 TO b%:Y%=RND(1000):NEXT
  340 FOR X%=1 TO 4000:U%=U%+2:Y%=(?U%)*
256+?(U%+1):Y%=Y%+RND(1000)
  350 MO%=Y% MOD 93:IF MO%=28 OR MO%=60
OR MO%=92 Y%=Y%+1
  360 ?U%=Y% DIV 256:?(U%+1)=Y% MOD 256
  370 NEXT X%
  380 IF sq%=1 GOTO 400
  390 *SAVE seq 5BCC 7B0B
  400 FOR X%=14500 TO 23499:?X%=126:NEXT
  410 CLS:PRINT"1. Encipher"'"2. Deciphe
r"'''"Select by keying 1 or 2"
  420 G$=GET$:IF G$<>"1" AND G$<>"2" GOT
O 420
  430 IF G$="1" PROCenciph ELSE PROCdeci
ph
  440 END
  450 :
 1000 DEF PROCenter(x%)
 1010 CLS:PRINT"Is message on disc? (Y/N
)"
 1020 IF FNYN="N" GOTO 1090
 1030 CLS:PRINT"Insert disc, then press
any key":G=GET
 1040 IF x%=2 GOTO 1070
 1050 *LOAD clear 4A38
 1060 ENDPROC
 1070 *LOAD crypt 4A38
 1080 ENDPROC
 1090 CLS:PRINT"Type message, ending wit
h ~":V%=&4A37
 1100 REPEAT
 1110 V%=V%+1:G%=GET:IF G%=127 V%=V%-2 E
LSE ?V%=G%
 1120 IF G%=13 PRINT ELSE PRINTCHR$(G%);
 1130 UNTIL G%=126
 1140 ENDPROC
 1150 :
 1160 DEF PROCoutput(x%)
 1170 A$(1)="encrypt":A$(2)="decrypt"
 1180 CLS:PRINT"Save "A$(x%)" to disc? (
Y/N)"
 1190 IF x%=2 PRINT"otherwise print"
 1200 IF FNYN="N" GOTO 1320
 1210 IF x%=2 N%=1:GOTO 1230
 1220 INPUT"How many copies? "N%
 1230 FOR M%=1 TO N%
 1240 PRINT"Insert disc, then press any
key":G%=GET
 1250 IF x%=1 F%=OPENOUT"crypt" ELSE F%=
OPENOUT"decrypt"
 1260 W%=&3A97
 1270 REPEAT
 1280 W%=W%+1
 1290 BPUT#F%,?W%
 1300 UNTIL ?W%=126
 1310 CLOSE#F%:NEXT M%
 1320 CLS:IF x%=2 GOTO 1340 ELSE PRINT"P
rint-out required? (Y/N)"
 1330 IF FNYN="N" ENDPROC
 1340 IF x%=2 N%=1:GOTO 1360
 1350 INPUT"How many copies "N%
 1360 VDU2:FOR M%=1 TO N%
 1370 W%=&3A97
 1380 REPEAT
 1390 W%=W%+1:IF x%=2 AND ?W%=123 PRINTC
HR$(32):GOTO 1420
 1400 IF x%=2 AND ?W%=125 PRINT:GOTO1420
 1410 PRINTCHR$(?W%);
 1420 UNTIL ?W%=126
 1430 CLS:PRINT"Paper - then press any k
ey":G=GET
 1440 NEXT M%
 1450 VDU3
 1460 ENDPROC
 1470 :
 1480 DEF PROCenciph
 1490 PROCenter(1)
 1500 U%=&5BCA:W%=&3A97:X%=0:V%=&4A37
 1510 REPEAT
 1520 V%=V%+1:G%=?V%
 1530 IF G%=126 GOTO 1640
 1540 IF G%=123 OR G%=125 GOTO 1520
 1550 IF G%=32 G%=123
 1560 IF G%=13 G%=125
 1570 IF G%=126 GOTO 1640
 1580 U%=U%+2
 1590 T%=256*(?U%)+?(U%+1)
 1600 H%=(G%+T%) MOD 93 +33
 1610 W%=W%+1:?W%=H%
 1620 X%=X%+1
 1630 IF X%=5 H%=32:W%=W%+1:?W%=H%:X%=0
 1640 UNTIL G%=126
 1650 PROCoutput(1)
 1660 ENDPROC
 1670 :
 1680 DEF PROCdeciph
 1690 PROCenter(2)
 1700 U%=&5BCA:W%=&3A97:V%=&4A37
 1710 Z%=U%
 1720 PRINT"Press SHIFT to advance displ
ay":PRINT:VDU14
 1730 REPEAT
 1740 V%=V%+1:G%=?V%
 1750 IF G%=126 GOTO 1890
 1760 IF G%<>32 GOTO 1800
 1770 U%=U%+10:Z%=U%
 1780 V%=V%+1:G%=?V%:IF G%=32 GOTO 1780
 1790 IF G%=126 GOTO 1890
 1800 Z%=Z%+2
 1810 T%=256*(?Z%)+?(Z%+1)
 1820 S%=66+T%-G%
 1830 IF S% MOD 93=0 H%=S% DIV 93 ELSE H
%=S% DIV 93 +1
 1840 H%=93*H%+G%-33-T%
 1850 IF H%=123 H%=32
 1860 IF H%=125 H%=13
 1870 IF H%=13 PRINT ELSE PRINTCHR$(H%);
 1880 W%=W%+1:?W%=H%
 1890 UNTIL G%=126
 1900 PRINT:PRINT:PRINT"Press space bar
to stop here, or any "'"other key to con
tinue":G%=GET:IF G%=32 END
 1910 PROCoutput(2)
 1920 ENDPROC
 1930 :
 1940 DEF FNYN
 1950 K%=GET AND 223:IF K%<>78 AND K%<>8
9 THEN 1950 ELSE K$=CHR$(K%):=K$
```

Listing 2

```
   10 REM Program Key
   20 REM Version B 1.0
   30 REM Author  Ruben Hadekel
   40 REM BEEBUG  Aug/Sep 1992
   50 REM Program subject to copyright
   60 :
  100 S%=0
  110 INPUT"Enter number "A$
  120 FOR N%=LEN(A$) TO 1 STEP -1
  130 B$=MID$(A$,N%,1)
  140 B%=VAL(B$)-1
  150 S%=B%*6^(LEN(A$)-N%)+S%
  160 NEXT
  170 PRINTS%
```

RISC USER

The Archimedes Magazine & Support Group

RISC User continues to enjoy the largest circulation of any subscription magazine devoted solely to the Acorn Archimedes range of computers. Its in-depth, authorative approach appeals to all users, whatever their interest and level of expertise, and the lively mixture of articles, programs, reviews and news is carefully selected to cater for all Archimedes owners from beginner to expert.

Existing BEEBUG members, who want to find out more about the Archimedes range, may either transfer their existing subscription to RISC User (at no extra charge), or extend their subscription to include both magazines.

A joint subscription will enable you to keep completely up-to-date with all innovations, and the latest information from Acorn and other developers for both the BBC micro and the Archimedes range. RISC User is the magazine for enthusiasts and professionals at all levels.

The Archimedes Magazine & Support Group

Here are some articles and series published in the most recent issues of RISC User:

NEURAL COMPUTING ON THE ARC

An introduction to this relatively new field in which hardware and software are made to simulate the working processes of neurons in the brain.

VECTOR

A review of this excellent object-oriented draw application from 4Mation.

DICTIONARY OF THE LIVING WORLD

A look at the exciting new "Multimedia encyclopedia of the Life Sciences" provided on a CD-ROM.

VIRTUAL REALITY EXPLORED

An entertaining article about the modern subject of virtual reality and a program exploring 3D images on the Archimedes.

TECH WRITER

A look at the technical word processor from Icon Technology, for generating text with mathematical equasions and diagrams.

TELEPHONE CALL COSTER

A useful application for monitoring the cost of telephone calls.

PC PAGES

A regular series devoted to the PC emulator and to the running of PC software on the Archimedes.

WIMP FUNCTION/PROCEDURE LIBRARY

A collection of functions and procedures (in four parts) providing all the necessary building blocks for your own Wimp programs.

WRITE-BACK

New readers' section of RISC User for comment, information, help - a magazine version of a bulletin board.

USING ANSI C

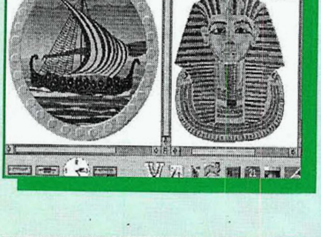
A series of articles on programming Desktop applications in C.

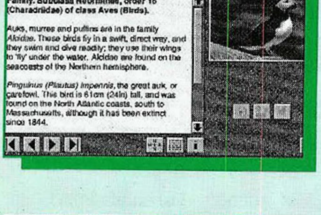
WP/DTP

A regular column on using different DTP and WP packages.

INTO THE ARC

A regular series for beginners.





Dictionary Of The Living World

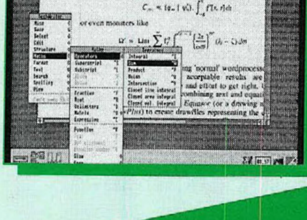
Alcidae

Auks, Murres, Puffins

Family. Subclass Neornithes, order 16 (Charadriidae) of class Aves (Birds).

Auks, murres and puffins are in the family *Alcidae*. These birds fly in a swift, direct way, and they swim and dive readily; they use their wings to 'fly' under the water. Alcidae are found on the seacoasts of the Northern hemisphere.

Pinguinus (Plautus) impennis, the great auk, or garefowl. This bird is 61cm (24in) tall, and was found on the North Atlantic coasts, south to Massachusetts, although it has been extinct since 1844.



Well, if you often have to produce documents which in
mathematical equations like

or even monsters like

SUBSCRIPTION DETAILS

As a member of BEEBUG you may extend
your subscription to include RISC User for only:

Destination	*Additional Cost*
UK,BFPO &Ch Is	£ 10.50
Rest of Europe and Eire	£ 15.40
Middle East	£ 19.60
Americas and Africa	£ 21.90
Elsewhere	£ 33.00

Dynamic Memory Window

This program by Sebastian Lazareno, originally published in BEEBUG Vol.4 No.5, displays a memory dump that is constantly updated.

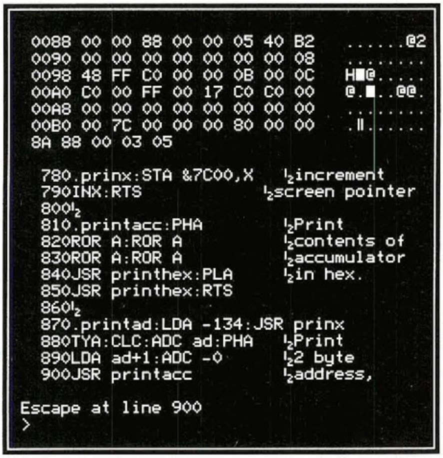
Many readers will appreciate the value of a static memory dump, giving a snapshot of the micro's memory, but such a display does not show changes in RAM as they occur. This program provides a dynamic memory window, allowing you to observe the Beeb or Master's memory in real time while using the keyboard, running other programs (including ROMs), or just 'resting'. The program can also display the contents of any sideways ROM, and is particularly useful for monitoring sideways RAM.

USING THE PROGRAM

Type the program in and save it before you run it, since an error may cause the computer to hang. When you run the program you will be asked for the start address of the display (i.e. to display zero page, enter 0).

The main routine extends through page 9 without intruding into page &A. It displays the contents of 48 bytes of memory in 'dump' format at the top of a mode 7 screen. You can scroll through memory using the vertical cursor keys with Shift (slow) or Ctrl-Shift (fast). The routine is called 50 times a second by the 'start of vertical sync' event. According to the Advanced User Guide (p.288) such routines should not last more than about 2 milliseconds. This one takes about 5.2 milliseconds, but problems are avoided as the full routine is called by only a proportion of the events generated.

The number of events per complete execution is initially set to 5 but can be altered using Shift along with the horizontal cursor keys. The ratio is shown at the bottom of the display, together with the number of events since the last execution and the display start address. The whole routine can be turned off using function key f7, and turned on using f8. This disables and enables the event generation.



```
0088 00 00 88 00 00 05 40 B2   ......@2
0090 00 00 00 00 00 00 00 08   ........
0098 48 FF C0 00 00 0B 00 0C   H█@.....
00A0 C0 00 FF 00 17 C0 C0 00   @.█..@@.
00A8 00 00 00 00 00 00 00 00   ........
00B0 00 7C 00 00 00 80 00 00   .‖......
8A 88 00 03 05

 780.prinx:STA &7C00,X   ½increment
 790INX:RTS             ½screen pointer
 800½
 810.printacc:PHA        ½Print
 820ROR A:ROR A          ½contents of
 830ROR A:ROR A          ½accumulator
 840JSR printhex:PLA     ½in hex.
 850JSR printhex:RTS
 860½
 870.printad:LDA -134:JSR prinx
 880TYA:CLC:ADC ad:PHA   ½Print
 890LDA ad+1:ADC -0      ½2 byte
 900JSR printacc         ½address,

Escape at line 900
>
```

Editing a Basic program while the memory display is active

POSSIBLE USES

1. ROMs. Look at locations starting at &8000 and type *HELP: by stopping the display with Shift-Ctrl you can explore any of your active ROMs. With a small modification to the program (see below) the contents of any sideways ROM or RAM can be continuously displayed.

2. Entering a line of Basic. Look at locations &700 onward (the Basic keyboard buffer), enter a line of Basic and see the line being tokenized when you press Return.

3. Keyboard entry. Still looking at &700 onward, use the Copy key to enter a few characters. Note that the characters entering the buffer are the ones you copied, and this is still so if Shift or Ctrl are pressed with Copy. Now look at &3E0 onward (the keyboard buffer). Note that when you Copy a character, the code entering the buffer is that for the Copy key, rather than the character copied, and that pressing Shift or Ctrl together with Copy causes a different code to enter the buffer. Now look at locations &EC and &ED (which contain 'current keys pressed' information) and see that pressing Copy produces yet another code (the 'INKEY' code), which is not affected by pressing Shift and Ctrl. These observations provide some clues as to how the Beeb processes keyboard input, and suggest further questions.

PROGRAM MODIFICATIONS

The program displays both Teletext character sets, i.e. ASCII 32-126 and 160-255, so many displayed characters are ambiguous. To display only the normal set delete line 670.

Sideways ROM/RAM can be selected with the following modifications. Change 'lock' to 'rom' in lines 390 and 410 and insert the following lines:

```
 150 rom=base+2:key=base+3:lock=base-1
 500 .start LDA rom:STA&FE30
 505 LDA#0:STA&308
1015 LDA&F4:STA&FE30
```

To inspect your sideways RAM (or ROM), set the display to start at &8000 and press Shift with the horizontal cursor keys to select the required ROM.

As it stands, the routine monopolises the event vector. The program could be modified to save the previous contents of the vector and exit via a JMP so that any other user routines using this event can take their turn.

PROGRAM NOTES

Lines 220-800: checks number of event, keys pressed, sets up pointers, text window.

Lines 810-1120: the display routine.

Lines 1130-1220: sets up soft keys. Clears buffer if necessary using equivalent of ON ERROR to detect full soft key buffer. Starts routine by invoking f8.

```
   10 REM Program MemWin
   20 REM Version B 0.7
   30 REM Author  S.Lazareno
   40 REM BEEBUG  Aug/Sept 1992
   50 REM Program Subject to Copyright
   60 :
  100 *FX13,4
  110 MODE7
  120 PRINT'"Press f7 to turn off, f8 to
 turn on"'"SHIFT + vertical cursors = SC
ROLL"'"SHIFT+CTRL+vert. cursors = fast S
CROLL"'"SHIFT+horiz.cursors = change sam
ple rate"
  130 ON ERROR GOTO 1600
  140 base=&8A:ad=base:INPUT"Start addre
ss &"S$:!ad=EVAL("&"+S$)
  150 key=base+2:lock=base+3
  160 line=base+4:code=&8C0:?key=0:?lock
=5
  170 REM EVENTV points to routine
  180 ?&220=code MOD &100:?&221=code DIV
 &100
  190 FORI%=0TO2STEP2
  200 P%=code
  210 [OPT I%:PHP:PHA:TXA:PHA:TYA:PHA
  220 LDX key:INX         \Increment
  230 STX key:CPX lock    \key,
  240 BEQ scrollcheck
  250 LDX #240:JMP end    \exit if < lock
  260 .scrollcheck        \is SHIFT or
  270 LDA &25A:TAX        \SHIFT LOCK
  280 AND #8:BNE ctrl     \pressed?
  290 TXA:AND #32
  300 BNE start           \If so, check
  310 .ctrl TXA:AND #&40\for CTRL,set
  320 LSR A:BNE scan      \acc. to alter
  330 LDA #8              \address pointer
  340 .scan LDX &EC        \Check whether
  350 BEQ skip             \a cursor
  360 PHA:SEC:TXA          \key has been
  370 SBC #128:TAX:PLA     \pressed,
  380 CPX #25:BNE right \if so,
  390 DEC lock:JMP start\take
  400 .right CPX #121      \appropriate
  410 BNE up:INC lock      \action.
  420 JMP start
  430 .up CPX #57:BNE down
  440 CLC:ADC ad:STA ad
  450 BCC skip:INC ad+1
  460 .skip JMP start
  470 .down CPX #41:BNE start
  480 EOR #&FF:ADC ad::STA ad
  490 BCS start:DEC ad+1
  500 .start LDA #0:STA &308
  510 LDA #24:STA &309   \Equivalent
  520 LDA #39:STA &30A   \to
  530 LDA #8:STA &30B    \VDU28,0,24,39,8
  540 ORA &D0:STA &D0    \Text window set
  550 LDA #1:STA line
  560 LDY #0:LDX #0:STX key\Print addr.
  570 .loopad:JSR printad  \and contents
  580 .loop:LDA (ad),Y      \of next 8
  590 JSR printacc          \locations,
  600 LDA #32:JSR prinx     \first in
  610 INY:ASL line:BCC loop\hex,
  620 ROL line:LDA #131:JSRprinx:INX
  630 TYA:SEC:SBC #8:TAY
  640 .loop2:LDA (ad),Y     \then in ASCI
I
  650 CMP #32:BCC dot       \if possible.
  660 CMP #127:BCC pound
  670 CMP #160:BCS next     \ Remove this
 line to prevent display of chars 160 -
255
  680 .dot LDA #ASC".":BNE next
  690 .pound:CMP#96:BNE hash \MODE 7
  700 LDA#35:BNE next        \display
  710 .hash CMP#35:BNE under \of
  720 LDA#95:BNE next        \these
  730 .under:CMP#95:BNE next \is
  740 LDA #96                \funny
  750 .next JSR prinx
  760 INY:ASL line          \Display 8
  770 BCC loop2:ROL line    \locations pe
r
  780 CPX #240:BCS end      \line,
  790 JMP loopad            \6 lines.
```

```
  800 \
  810 .printhex:AND #&F     \Print
  820 CLC:ADC #&30:CMP #&3A\digit in
  830 BCC prinx:ADC #6      \hex
  840 .prinx:STA &7C00,X    \increment
  850 INX:RTS               \screen point
er
  860 \
  870 .printacc:PHA         \Print
  880 ROR A:ROR A           \contents of
  890 ROR A:ROR A           \accumulator
  900 JSR printhex:PLA      \in hex.
  910 JSR printhex:RTS
  920 \
  930 .printad:LDA #134:JSR prinx
  940 TYA:CLC:ADC ad:PHA    \Print
  950 LDA ad+1:ADC #0       \2 byte
  960 JSR printacc          \address,
  970 PLA:JSR printacc      \increment
  980 LDA #135:JSR prinx    \screen
  990 RTS                   \pointer.
 1000 \
 1010 .end
 1020 LDA #133:JSR prinx    \This routine
,
 1030 LDA #base             \executed
 1040 JSR printacc          \at every
 1050 LDA #135:JSR prinx    \call, prints
 1060 LDY #0                \the start
 1070 .loop:LDA base,Y      \address
 1080 JSR printacc          \of the
 1090 LDA #32:JSR prinx     \display, and
 1100 INY:CPY #4:BNE loop   \the contents
 1110 PLA:TAY:PLA:TAX       \of lock & ke
y
 1120 PLA:PLP:RTS           \then exits.
 1130 .invoke
 1140 OPT FNonerror
 1150 OPT FNturnoff         \Set f7
 1160 OPT FNturnon          \Set f8
 1170 OPT FNbreak           \Set f10
 1180 LDA line:STA&202      \ON ERROR
 1190 LDAline+1:STA&203     \OFF
 1200 LDA#138:LDX#0         \Invoke f8 to
 1210 LDY#136:JSR&FFF4      \turn on rout
ine
 1220 RTS:]
 1230 NEXT
```

Continued on page 48

Mr Toad's Machine Code Corner

Direct from the BEEBUG lily pad, Mr Toad tells us why Hex is good for us.

Hello, Toad fans. This month we begin with a health warning; it has come to our notice that certain frivolous individuals have been referring to MCC as 'Frog Forum'. If Mr T catches them, they'll get a smack in the griblets; there is no place for irreverent pseudo-humour in a serious field like assembler programming.

"In previous issues," writes Ossian de Lytelpubes of Bognor, "Mr T has been extremely rude about people who use decimal in assembler. Is hex really so important?" Well, Mr de Lytelpubes, we think so. Old hands like yourself may have fixed opinions on this score, but perhaps we can persuade a few newcomers to 6502. Here goes...

In a word, hexadecimal numbers have a one-to-one correspondence with binary 'nibbles' (four-bit numbers), and that's it and all about it. What does that mean to yer average amphibian in the street, though?

SOME THINGS IT MEANS

Firstly, in hex you can stick two two-figure numbers together and get the correct four-figure value. An easy example is 32769, &8001 in hex, which we'll say is an address. To put 32769 into (say) X and Y, as when preparing for a call to OSWORD you have to write LDX #32769 MOD 256:LDY #32769 DIV 256, or else work out the sums and write LDX #1:LDY #128. Using hex, we simply write LDX #01:LDY #&80.

It's only part of the story, though. Here's the real issue: going from bit 7 down to bit 0, the decimal values of each bit of a byte are: 128, 64, 32, 16, 8, 4, 2, 1. Not very memorable. In hex they go &80, &40, &20, &10, 8, 4, 2, 1 (even a dyed-in-the-pond hex freak like Mr T doesn't put '&' before numbers 0 to 9. I have actually seen '&0' in print, but that's going too far). So in each four-bit nibble, we have values 8, 4, 2 and 1, and you chain these values together to make whole bytes, usque ad infinitum. Any bit is worth &10 times the bit 4 places to the right of it. The high nibble of a single byte or of the low byte of a multi-byte number (e.g. &FF) has bits valued &80, &40, &20, &10; the low nibble of the high byte of a two-byte number (e.g. &FFFF) has bits worth &800, &400, &200, &100, the bits of its high nibble (e.g. &FFFF) are worth &8000, &4000 and so on and so on until the cows come home.

With this principle you can actually visualise the binary version of a byte, thinking always of 8, 4, 2, 1. At first you may need to jot down &80, &40.....1 on paper and write in the ones and noughts underneath, but soon you'll be doing them in your head. You could make a table and stick it to the chimney of your Beeb:

```
&F 1111
&E 1110
&D 1101
&C 1100
 .
 .
 .
2 0010
1 0001
0 0000
```

Best not to, though - you'll get to rely on it, and never learn.

Always split the byte into nibbles and think of '8, 4, 2, 1'. Since these four numbers add up to &F there is never a carry to the next nibble. If the byte is 10101100, first take 1100: that's 8+4=&C. Write that down or remember it. The

1010 is 8+2=&A - times &10, of course - 10101100 is therefore &AC. Or, since there is no carry, you can equally well work from left to right and take the high nibble first; whatever turns you on. Now you can't do true addition from left to right in hex or any other base, because of the possibility of a carry - it's just in converting binary to hex that a carry never occurs.

In decimal, the only way to do it is this:

```
bit 7 128
bit 5  32
bit 3   8
bit 2   4
      172    i.e. &AC
```

The answer 172 is for the byte as a whole: the two nibbles, 160+12, are 'mixed up' in the one number - in fact, a ten has been carried from the value of the low nibble to the high.

GOING BITWISE

Let's consider bitwise operations. We'll try manipulating bit 5 of a byte. As I implied above, the bits are conventionally numbered from bit 7 on the left, worth &80, down to bit 0 on the right, worth 1. Quick count on fingers: "8, 4, 2..." bit 5 is worth &20. To set it, therefore, write ORA #&20. Admittedly, ORA #32 is not all that much harder to remember and even saves a byte in the assembly-text. To reset bit 5, though, AND #&DF is a lot easier than working out AND #223. It's so easy to do the subtraction &FF-&20 mentally by counting down: &FF,&EF,&DF. You could write AND #(255-32), but that's an abject cop-out - you'd never perpetrate such an outrage, would you, gentle reader? Mr T never has. Ahem... In practice, one commonly does such bitwise operations with several bits at once; you can't really visualise such things except in binary, and that can only be done easily from hex.

A CAPITAL IDEA

Incidentally, why all this toggling of bit 5? What it does is shift the letters A - Z and a - z between upper and lower case. Say your program has just displayed a menu and there are several choices, each requiring one keypress. Do you test for a capital or a lower case letter? You should really allow both, or users may get their knickers in a twist. You could write:

```
CMP #ASC"C":BEQ croak
CMP #ASC"c":BEQ croak
```

but unless you're testing for only one key, there's a neater way.

The difference between a capital letter and its lower case equivalent is &20 - i.e. bit 5 of its binary form. The small letters are ASCII 011xxxxx and the capitals are 010xxxxx. Therefore if you write AND #&DF after the call to OSRDCH, it forces upper case, and, although one rarely sees it used, ORA #&20 forces a small letter. Since all the numerals are 001xxxxx, the latter allows numerals through unaltered, whereas AND #&DF masks them off to 000xxxxx, which means you need a test and branch beforehand to skip over the AND #&DF when a numeral comes up (CMP #ASC "A":BCC label). If your input routine has to cope with both numerals and letters, therefore, this is the better way.

```
JSR osrdch
ORA #&20
CMP #ASC"c"
etc. etc.
```

But I digress. The point is that such bitwise operations are very common, and that they are only easily manageable from hex.

OK, I hear you cry (well, more of a sort of a grobbly yawning noise, really,) why not have the best of both worlds? Use hex for the nasty ones and when the book happens to quote hex, otherwise use decimal. Your rabidly conservative reptile has three objections.

1) It's not very professional, and yes, that is a serious argument. Programming is like wiring - it usually works even if it's untidy and sloppily laid out, but in the end you get more trouble than you save.

2) You must use hex all the time if you're ever going to get good at it. It's got to become as automatic as decimal - well, maybe not quite, since you've been using decimal for so long, but it must become pretty well ingrained.

3) Being a programmer is a bit like being a lawyer, only useful; how are we going to con the mugs into grovelling to us megabrains if we use numbers they can understand?

This month's competition: arrange the above arguments in order of importance. Wrong entries get you two tickets to the BEEBUG Winter Ball, which is held in the front showroom and has been known to last several minutes. The best letter will be rewarded with the customary "I'M A SWOT" badge.

I hope I've answered your question, Mr de Lytelpubes. I did write to you personally, but absent-mindedly addressed the envelope to another regular, Ossian de Bognor of Littlehampton. That's it for now, reptile-readers. Next month: sweeping the flue of the Acorn Electron, and a review of 23 brands of steam-coal for the B+.

AND FINALLY...

For those of you still struggling with last month's limerick the last two lines should be read:

"Was cos theta times eight,
Over x plus the square root of damn all."

Dynamic Memory Window (continued from page 45)

```
 1240 CALLinvoke:END
 1250 DEF FNonerror:pass=P%
 1260 FORopt=0TO2STEP2:P%=pass
 1270 [OPT opt
 1280 LDA&202:STA line
 1290 LDA&203:STA line+1
 1300 LDA #(onerr MOD 256):STA &202
 1310 LDA #(onerr DIV 256):STA &203
 1320 BNE setup
 1330 .onerr:LDA#18:JSR&FFF4
 1340 .setup:]
 1350 NEXT:=I%
 1360 :
 1370 DEFFNturnoff:str$="K.7*FX13,4|M":=
FNoscli
 1380 :
 1390 DEFFNturnon:str$="K.8MO.7:V.31,0,8
:*FX14,4|M":=FNoscli
 1400 :
 1410 DEFFNbreak:str$="K.10O.|M?&220=&"+
STR$~(code MOD256)+":?&221=&"+STR$~(code
 DIV256)+"|M":=FNoscli
```

```
 1420 :
 1430 DEFFNoscli
 1440 pass=P%:FORopt=0 TO2 STEP2
 1450 P%=pass
 1460 [OPT opt
 1470 LDX#(string MOD256)
 1480 LDY#(string DIV256)
 1490 BNEoscli
 1500 .string
 1510 ]
 1520 $P%=str$
 1530 P%=P%+LEN$P%+1
 1540 [OPT opt
 1550 .oscli JSR&FFF7
 1560 ]
 1570 NEXT
 1580 =I%
 1590 :
 1600 ON ERROR OFF:MODE 7
 1610 IF ERR=17 END
 1620 REPORT:PRINT" at line ";ERL
 1630 END
```

Linked Lists (Part 2)

Mike Williams explores the world of linked lists.

Last month I introduced you to the idea of a linked list, using two arrays for its implementation, one to hold the items of data, and one to hold the pointers which link the elements of the list in some preconceived order.

In the interests of providing a working program I was unable to devote much of last month's article to its operation, or to some of the problems which arise in coding. So let's make amends now.

If you will recall, I postulated four principal routines which would be needed:

```
PROCadd_element(data)
FNdelete_element(data)
FNfind_element(data)
FNdisplay_data
```

The first two provide the means by which data elements can be added to and removed from the linked list. The other two perform functions which we might need to carry out on a linked list: finding an element in the list (or determining its absence), and displaying in order the elements in the list.

IMPLEMENTING A FREE LIST

Every time an element is added to the current data list we need to find an empty element into which we can place the data, and then insert the new data element into the current list. One way of doing this is to use the concept of a *free list*. Initially, all the empty elements are chained together as a list. In the arrays, element zero is chained to element one, which is chained to element two and so on. A variable called *free* points to the head of the free list, initially element zero. When a data item is to be added to the current data list, we use the free list to locate the first available free element, resetting the *free* pointer to the next free element in the chain. Whenever an element is deleted from the current data list, the element is added to the head of the free list, and the *free* pointer adjusted accordingly. Thus we can take elements from the free list, and return discarded elements to it. In last month's program, the free list was initialised in line 120.

Note that although we refer to a free list and to a data list, both are part of the same two arrays, *Data()* and *Link()*. In effect the two lists are separate but intertwined (see figure 1). In the same way it is perfectly possible to have several distinct data lists all interleaved within the same array implementation.

The end of a list is indicated by a so-called *null pointer*, and in our program this is indicated by a pointer value of -1, since all genuine array positions (elements) are in position zero or greater. For the data list, the pointer to the first element (held in the variable *head*) is therefore set to -1 initially to indicate an empty list.

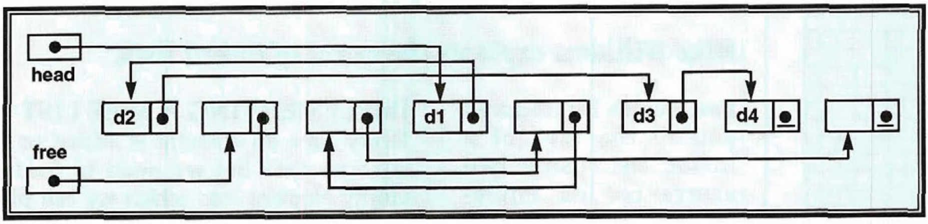


Figure 1. Interleaved data and free lists

ADDING AN ELEMENT TO A LINKED LIST

Coping with the addition of elements within a linked list is, in principal, straightforward. We first locate a free element. Then start at the head of the data list and follow the links until we find the first element whose data value is less than the new element (the ordering I have selected for the demo). We now insert the new element between this one and its predecessor.

The coding has to maintain two pointers all the time, such that when the correct position for the new element has been found, the first pointer points to the preceding element, and the second pointer to the following element (call them elements L1 and L2). Suppose that our new data element is at position 'L' in the array. To insert the new element we make the link of L1 point to L, and the link of L point to L2 (a copy of the original link from L1). That's all there is to it (see figure 2).

In practice, a number of special cases make life much more complex. Because the head of the current list is indicated by the variable head, rather than an element of *Link()*, both the case of an empty list, and the case where the new element becomes the first element of the current list, must be treated separately as both involve explicit references to *head*. In the procedure *PROCadd_element* these special cases are dealt with in lines 1030 and 1050 respectively. For convenience, the coding of the procedure PROCadd_element is repeated below.

```
1000 DEF PROCadd_element(data)
1010 LOCAL L,L1,L2,f%:f%=FALSE
1020  L=free:free=Link(free):Data(L)=data
1030 IF head=-1 THEN Link(L)=head:
     head=L:ENDPROC
1040 L1=head:L2=L1
1050 IF data<Data(L1) THEN Link(L)=head:
     head=L:ENDPROC
1060 REPEAT
1070 L1=L2:L2=Link(L2)
1080 IF L2=-1 THEN f%=TRUE ELSE
     IF data<Data(L2) THEN f%=TRUE
1090 UNTIL f%
1100 Link(L)=L2
1110 Link(L1)=L
1120 ENDPROC
```

The other special case to be considered is when we reach the end of the current list without satisfying the condition of finding an existing data element less than the new one. In this case the new element becomes the last element in the list, and applying the standard insertion steps causes the null pointer in the previous last element to be copied into the link position of the new last element. The REPEAT-UNTIL loop in lines 1060 to 1090 thus has two possible terminating conditions: the appropriate position in the linked list has been found, or the end of the list has been reached.

Implementing this is, even so, trickier than it might seem. Look at the coding in lines 1060 to 1090 again. You might think it would have been simpler to have written:

```
1060 REPEAT
1070  L1=L2:L2=Link(L2)
1080 UNTIL L2=-1 OR data<Data(L2)
```

However, although we can see that the two conditions are mutually exclusive, and that if the first condition is TRUE, then the second condition is redundant, that is not how BBC Basic works. Instead, it evaluates both conditions to a logical TRUE or FALSE, and then applies the OR operator to contain the final result. In the event that the end of the list has been reached (and L2=-1), trying to evaluate Data(L2) will give the error message "Subscript out of range".

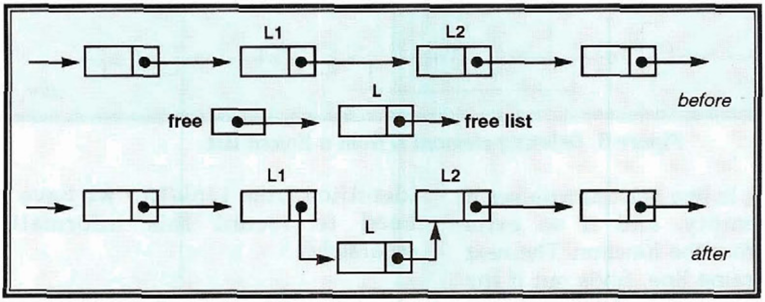


Figure 2. Adding element L between L1 and L2

The solution is to use a multiple IF-THEN construction as shown in line 1080. The part of the statement involving Data(L2) is only evaluated if the first condition (L2=-1) proves to be FALSE, and thus no error now arises when L2=-1. This is an excellent example of where a detailed and precise knowledge of the workings of Basic is essential if the coding is to perform as we wish.

There are also other considerations which should be taken into account if a robust and fullproof implementation is to be achieved (although I must confess to ignoring these considerations in last month's demo). Since we are implementing our linked list via the use of two arrays *Data()* and *Link()* dimensioned at the start of the program, there is a finite limit to the total number of elements which can be accommodated. The attempt to obtain the first free element in the free list might founder because the free list is empty. This would need checking and reporting. In more sophisticated implementations, the allocation of memory for this purpose would be more flexible, so that all of available memory (at that time) could be used. This would be more easily accomplished using indirection operators rather than Basic's array notation.

DELETING AN ELEMENT

Deleting an element from a list has many similarities with the foregoing, and in some ways is its mirror image. The routine is coded as a function so that it can return the position of the data element to be deleted if this is found, or a -1 if the data element cannot be located. Again, the coding included in last month's program is repeated here for convenience.

```
1200 DEF FNdelete_element(data)
1210 LOCAL f%,f1%,L,L1:L=head:L1=L:
     f%=FALSE:f1%=FALSE
1220 IF L=-1 THEN =L ELSE
     IF Data(L)=data THEN head=Link(L):
     Link(L)=free:
```

```
     free=L:=L
1230 REPEAT
1240 L1=L:L=Link(L)
1250 IF L=-1 THEN f%=TRUE ELSE
     IF Data(L)=data THEN f1%=TRUE
1260 UNTIL f% OR f1%
1270 IF f1% THEN Link(L1)=Link(L):
     Link(L)=free:
     free=L
1280 =L
```

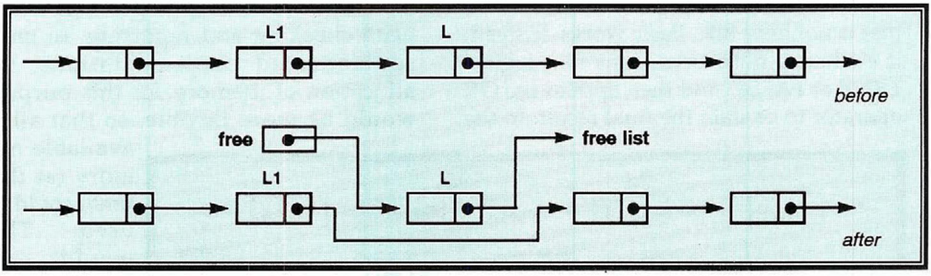


Figure 3. Deleting element L from a linked list.

The first check, in line 1220, determines if the list is empty, and if so exits immediately from the function. The next check, in the same line, finds out if the sought data element is the first in the linked list, in which case deletion involves explicit reference to the list head pointer. Lines 1230 to 1260 then form the main loop of the function wherein the program follows the links of the chained data elements until the data element being sought is found, or the end of the list is reached. Notice, in line 1250, a similar nested IF-THEN construction to that used in PROCadd_element in order to test correctly for the two possible, but mutually exclusive, terminating conditions.

To remove an element from a linked list, the link pointer of the preceding element has to be changed to point to the following element (by copying the pointer from the element to be deleted), and the now released element is added to the start of the free list for further use (see figure 3). There is one further, if minor point of interest: in contrast with the previous routine, we need this time only to record the position of the preceding element, and the position of the element to be deleted (equivalent to L1 and L in our previous notation). Because L is initially part of the chain, the following element can always be identified from Link(L) - we have no need to record this information separately.

The other two routines coded last month, to located a data element, and to display a list of data elements, are both much simpler, since there is no need to manipulate any pointers at all - we just use them. Even the special cases we have referred to previously cause far fewer problems. As a result, I do not propose to discuss these or the rest of the demo program in detail. Careful examination of the two routines detailed above, together with a reading of the whole program should suffice to make matters clear.

That explanation has used up all our space for this month. Next time we will take a look at more complex forms of linked list structure, such as doubly linked lists. For convenience, the full demo program, *List01*, is repeated on this month's magazine disc. B

ColourFlash

David Cox gives you flashing coloured text, and probably a headache.

The machine code utility in the listing below, with its attached demo program, shows how you can get flashing text in all modes except mode 7 which has its own inbuilt flashing features. The text doesn't just flash, it cycles through the colours highlighting the text very effectively. It also affects graphics in the same way.

HOW IT WORKS

The code is event-driven using the vertical sync event to interrupt whatever else is going on in the machine and change the colour of all the text and graphics on the screen printed in logical colour 1 using VDU 19. You could change the designated colour by changing the variable *colour* at line 1030.

USING IT

To get flashing text, type in the program and run it. The demo display uses mode 1. Once the code is assembled, any use of colour 1 in any mode will give you flashing text. Note, you must only run the code once or the machine will lock up.

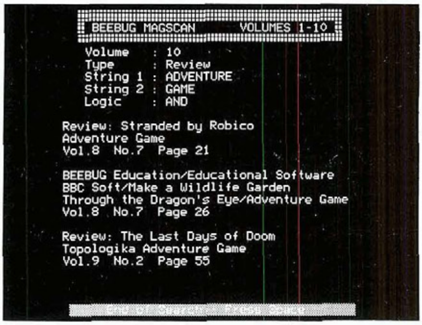
In modes with more than two colours, all the other colours are available as normal but if you use it in modes 0, 3, 4 or 6 then all the text will be flashing. Have fun!

```
   10 REM Program ColourFlash
   20 REM Version B 1.0
   30 REM Author  David.C.Cox
   40 REM BEEBUG  Aug/Sept 1992
   50 REM Program subject to copyright
   60 :
  100 MODE 1
  110 PROCSetUp
  120 PROCAssemble
  130 PROCDisplay
  140 END
  150 :
 1000 DEF PROCSetUp
 1010 VDU23;8202;0;0;0;
 1020 DIM prog% &100
 1030 colour=1
 1040 event=&220
 1050 osbyte=&FFF4
 1060 oswrch=&FFEE
 1070 DIM message$(4)
 1080 RESTORE 1870
 1090 FORloop%=1TO4
 1100 READmessage$(loop%)
 1110 NEXT
 1120 ENDPROC
 1130 :
 1140 DEF PROCDisplay
 1150 COLOUR colour
 1160 FORloop%=1TO4
 1170 PRINTTAB(20-LEN(message$(loop%))/2
,loop%*2);message$(loop%)
 1180 NEXT
 1190 COLOUR 3
 1200 PRINTTAB(5,11);"This program is EV
ENT driven."
 1210 ENDPROC
 1220 :
 1230 DEF PROCAssemble
 1240 FOR ION=0 TO 2 STEP 2
 1250 P%=&900
 1260 [OPTION
 1270 .colours
 1280 LDY #0
 1290 STY &70
 1300 STY &72
 1310 LDA #inter MOD 256
 1320 STA event
 1330 LDA #inter DIV 256
 1340 STA event+1
 1350 LDA #14
 1360 LDX #4
 1370 LDY #0
 1380 JSR osbyte
 1390 RTS
 1400 RTS
 1410 .colnum
1420 EQUB 1
1430 EQUB 2
1440 EQUB 3
1450 EQUB 7
1460 EQUB 3
1470 EQUB 2
1480 .inter
1490 PHA:TXA:PHA:TYA:PHA
1500 LDA &72
1510 CMP #5
1520 BNE exit
1530 LDY &70
1540 LDA colnum,Y
1550 STA &71
1560 LDA #19
1570 JSR oswrch
1580 LDA #colour
1590 JSR oswrch
1600 LDA &71
1610 JSR &FFEE
1620 LDA #0:STA &72
1630 JSR oswrch
1640 JSR oswrch
1650 JSR oswrch
1660 JSR oswrch
1670 JSR oswrch
1680 LDY &70
1690 CPY #5
1700 BEQ resetcol
1710 INY
1720 STY &70
1730 .exit
1740 INC &72
1750 PLA:TAY:PLA:TAX:PLA
1760 RTS
1770 .resetcol
1780 LDY #0
1790 STY &70
1800 JMP exit
1810 RTS
1820 ]
1830 NEXT
1840 CALL colours
1850 ENDPROC
1860 :
1870 DATA ColourFlash,By David.C.Cox,v.
B 1.0 (BEEBUG-Aug/Sept-92).,What do you
think?
```

Magscan

Comprehensive Magazine Database for the BBC Micro and the Master 128

An updated version of Magscan, which contains the complete indexes to all BEEBUG magazines from Volume 1 Issue 1 to Volume 10 Issue 5



```
BEEBUG MAGSCAN      VOLUMES 1-10
 Volume   : 10
 Type     : Review
 String 1 : ADVENTURE
 String 2 : GAME
 Logic    : AND

Review: Stranded by Robico
Adventure Game
Vol.8  No.7  Page 21

BEEBUG Education/Educational Software
BBC Soft/Make a Wildlife Garden
Through the Dragon's Eye/Adventure Game
Vol.8  No.7  Page 26

Review: The Last Days of Doom
Topologika Adventure Game
Vol.9  No.2  Page 55

End of Search - Press Space
```

Magscan with disc and manual **£9.95**+p&p

Stock codes:	0005a	5.25"disc 40 track DFS
	0006a	5.25"disc 80 track DFS
	1457a	3.5" ADFS disc

Magscan update **£4.75** +p&p

Stock codes:	0011a	5.25"disc 40 track DFS
	0010a	5.25"disc 80 track DFS
	1458a	3.5" ADFS disc

Magscan allows you to locate instantly all references to any chosen subject mentioned anywhere in the 95 issues of BEEBUG magazine to date.

Just type in one or two descriptive words (using AND/OR logic), and you can find any article or program you need, together with a brief description and reference to the volume, issue and page numbers. You can also perform a search by article type and/or volume number.

The Magscan database can be easily updated to include future magazines. Annual updates are available from BEEBUG for existing Magscan users.

Some of the features Magscan offers include:

- full access to all BEEBUG magazines
- rapid keyword search
- flexible search by volume number, article type and up to two keywords
- keyword entry with selectable AND/OR logic
- extensive on-screen help
- hard copy option
- easily updatable to include future magazines
- yearly updates available from BEEBUG

RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Tel (0727) 40303 Fax (0727) 860263

Public Domain Software

Alan Blundell gets more discriminating and compiles his own 'best of PD' collection.

It's interesting to note how attitudes to PD software for the Beeb have changed over the last couple of years. Two years ago, there may have been a fair amount of software available via bulletin boards (mostly music, of which more later) but users without a modem saw little of it. When the first few discs of submissions to PD libraries were compiled, people enthused over the idea. I received letters saying 'don't let the Beeb die!' or 'this is just what we need to keep interest in the BBC alive'. Now that there are over 40 megabytes of PD commonly available, I find that expectations have increased beyond the simple pleasures of finding out about a 'new' kind of software.

Like most subjects, PD software comes in very good and not so good standards, with most of what is available coming somewhere in the middle. The lower and middle ground is naturally less of interest when there is something at the 'very good' end of the range to suit your needs, and this leads some people to comment unfavourably on software which, two years ago, received much more favourable reactions. This is a natural reaction, which parallels the growth of PC public domain software: nowadays, a PC shareware program is expected to have all of the features of an equivalent commercial package. Some do, so others either have to have these features or fade into obscurity. Of course, part of this sophistication is the success of shareware on the PC - a good programmer can make money without having marketing skills or costs, by writing a good program and unleashing it on the untold millions of PC users. BBC programmers are now a lot thinner on the ground - the sad fact is that many of the best programmers have moved on to other, newer machines.

Still, I regularly receive copies of programs which I keep for my own use because they do their job well. This month I will note a few programs which I think are amongst the best of what PD has to offer. The best programs are those which you want because they do something that you want to be done and they do it well, whether they are utilities for file conversion, games, clip art, educational programs or whatever.

I read in a magazine recently that 'all the best adventure games are on the BBC'. If you like graphically illustrated adventures, I'm not so sure that that is true, but for my taste in more traditional games which let you use your imagination, there are some excellent games available. The Acornsoft series of adventures contained some of the best original games, so I was very pleased to hear from Derek Haslam, who wrote 'Gateway to Karos', which was published by Acornsoft. Derek has written a sequel called 'Mirror of Khoronz', which is going to stay on my hard disc until I've finished it. This sequel is a multi-user game and has some quite novel features (as well as using almost every byte of RAM available in the machine!). It does what I want it to do - entertain me - very well. I'm not going to say any more about it now - if you like adventure games, find a copy!

There are other good quality adventures available in the public domain, including Alpine Software's 'Plague Planet' (shareware) and Ian Hudson's 'Insomnia' and 'I Spy Red'. Arcade style games on the other hand, as I've written before, seem to be one programming challenge that everyone and his dog have had a go at. The results aren't always as entertaining for others as they may be for the writer. Although there are several I could mention, one game in particular has recently redefined my expectations of PD games - 'Snacker', by the mysteriously named Alt. Snacker is one of PacMan's relatives, not a new idea as Alt would be the first to admit. This implementation though, to my taste at least, is simply the best version of the game that I've played on a BBC.

On to more serious matters. I am pleased to report a success in my campaign to get commercial software which is no longer actively marketed re-released as PD or as shareware (that isn't actually true, as the author had re-released the program as PD before I ever got in touch with him to ask if I could distribute it - but it does help the cause). Many BBC owners of longer standing will remember or even have a copy of Howard Spurr's excellent protected disc copying program, 'Disc Duplicator III' for the BBC B. Whilst never intended as an aid to piracy, 'DDIII' is a handy way of insuring against corruption of your protected commercial software discs by making a backup copy.

The program works with the 8271 disc controller chip used in the BBC B but not with the 1770 chip used in the Master. Howard told me that he had decided to re-release it as PD because, when he researched the 1770 chip, he found too many ways in which discs could be protected which could not be got round by a copy program, so he could no longer guarantee to support the program as fully. If you have a BBC B with an old 8271 disc system and a need to backup those precious master discs, this is the program to have.

David Harper's SWAP ROM has never been sold commercially, but is nevertheless a very well produced piece of software for the Master series. It only performs one task, but does it excellently; it transfers programs in Basic into Master Edit text files in memory, and gets round the limitations of the Master's built-in Basic command EDIT by using sideways RAM as workspace. It also strips/inserts line numbers during the process, so forcing the use of structured programming style. In frequent use, I have found it fast and convenient and I wouldn't readily do without it now.

One person who has done more for PD on the Beeb than most is Alan Phillips of Lancaster University. His teletext ROM and macro assembler system are as good as you will get anywhere, and both are models of how good PD software should be presented - good at doing what they set out to do and well documented. These and the programs mentioned above, together with others mentioned in previous columns, are amongst my collection of programs for my own use, and I see no need to look for commercial alternatives. This is not intended to be a comprehensive list of all the worthwhile PD software in any way; they are simply the ones that are most useful to me, for my own interests.

OTHER MATTERS...

The subject of music programs and sampled soundtracks is one which I have been thinking about, following certain events which have recently affected ST and Amiga PD libraries. I will relate that story in the next column, once I know the situation more clearly. For now, I'd like to appeal for help in solving a mystery: if you own/use a Master 128, does your Acorn Welcome disc have a file called *Spriter* in directory *$.Library?* If it does (or even if it doesn't), I would be grateful to hear from you and to know the age of your micro and/or its serial number. The reason will become clear when I describe in a future column the mystery which is confusing me.

```
 2690 SOUND1,S%,45+L%*8,2
 2700 NEXT
 2710 ENDPROC
 2720 :
 2730 DEF PROCembs:t%=0:F%=0
 2740 FOR Z%=1 TO 3
 2750 IF D%(Z%)=L%t%=t%+V%(Z%):F%=F%*F%+
1
 2760 NEXT
 2770 t%=t%*5*F%:SC%(L%)=t%
 2780 ENDPROC
 2790 :
 2800 DEF PROCrun
 2810 IF D%(1)=D%(2) AND D%(1)=D%(3) F%=
5 ELSE F%=1
 2820 IF V%(1)<>V%(2) AND V%(1)<>V%(3) A
ND V%(2)<>V%(3) SC%(L%)=40*F%
 2830 ENDPROC
 2840 :
 2850 DEF PROCpair:t%=0
 2860 IF D%(1)=D%(2) t%=t%+50
 2870 IF D%(1)=D%(3) t%=t%+50
 2880 IF D%(2)=D%(3) t%=t%+50
 2890 IF V%(1)=V%(2) t%=t%+20
 2900 IF V%(1)=V%(3) t%=t%+20
 2910 IF V%(2)=V%(3) t%=t%+20
 2920 SC%(L%)=t%
 2930 ENDPROC
 2940 :
 2950 DEF PROCmatch
 2960 FOR J%=1 TO 3
 2970 IF C%(J%)=MC% F%=4 ELSE F%=1
 2980 IF D%(J%)=MD% SC%(9)=SC%(9)+V%(J%)
*20*F%
 2990 NEXT
 3000 ENDPROC
 3010 :
 3020 DEF PROCmix
 3030 c%=0:d%=0:v%=0
 3040 IF C%(1)<>C%(2) AND C%(1)<>C%(3) A
ND C%(2)<>C%(3) THEN c%=-1:SC%(L%)=SC%(L
%)+20
 3050 IF D%(1)<>D%(2) AND D%(1)<>D%(3) A
ND D%(2)<>D%(3) THEN d%=-1:SC%(L%)=SC%(L
%)+20
 3060 IF V%(1)<>V%(2) AND V%(1)<>V%(3) A
ND V%(2)<>V%(3) THEN v%=-1:SC%(L%)=SC%(L
%)+20
 3070 IF v% AND c% THEN SC%(L%)=SC%(L%)+
30
 3080 IF v% AND d% THEN SC%(L%)=SC%(L%)+
30
 3090 IF c% AND d% THEN SC%(L%)=SC%(L%)+
30
 3100 IF D%(1)<>MD% AND D%(2)<>MD% AND D
%(3)<>MD% AND d% THEN SC%(L%)=SC%(L%)*2
 3110 ENDPROC
 3120 :
 3130 DEF PROCtoptbl
 3140 FOR J%=0 TO 9:PRINTrbk$:NEXT
 3150 FOR J%=10 TO 20:PRINTbbk$:NEXT
 3160 FOR J%=21 TO 24:PRINTrbk$:NEXT
 3170 PRINTTAB(5,11)"TOP SCORE TABLE";
 3180 IF game%=3 THEN tbon%=(T%-tl%)*3 E
LSE tbon%=(T%-tl%)*0.1
 3190 IF tbon%<0 tbon%=0
 3200 PRINTTAB(4,3)"Hand score"SC%(11)-S
C%(10)
 3210 SC%(11)=SC%(11)+tbon%
 3220 PRINTTAB(4,4)"Time bonus"tbon%
 3230 PRINTTAB(4,5)"Game bonus"SC%(10)
 3240 PRINTTAB(9,7)"Total"SC%(11)
 3250 FOR J%=1 TO 7
 3260 IF SC%(11)>P%(J%) THEN P%=J%:J%=7
 3270 NEXT
 3280 FOR J%=1 TO 6
 3290 PRINTTAB(4,J%+12)P$(J%),P%(J%)
 3300 NEXT
 3310 IF P%<7 PROCtopsc
 3320 FOR J%=20 TO 21:PRINTTAB(4,J%)SPC(
35);:NEXT
 3330 PRINTTAB(4,22)"Another game? Y/N";
 3340 *FX15,0
 3350 Q$=GET$
 3360 VDU19,1,1;0;
 3370 IF Q$="N" END
 3380 ENDPROC
 3390 :
 3400 DEF PROCshuffle
 3410 FOR J%= 6 TO P% STEP -1
 3420 P%(J%)=P%(J%-1):P$(J%)=P$(J%-1)
 3430 NEXT
 3440 P%(P%)=SC%(11):P$(P%)=name$
 3450 FOR J%=P% TO 6
 3460 PRINTTAB(4,J%+12);SPC(20)
 3470 PRINTTAB(4,J%+12)P$(J%),P%(J%)
 3480 NEXT
 3490 ENDPROC
 3500 :
 3510 DEF PROCtopsc
 3520 PROCping
 3530 PRINTTAB(4,20)"Enter your name for
 inclusion"
 3540 PRINTTAB(4,21)"in the TOP SCORES T
ABLE"
 3550 INPUTTAB(4,22)name$;
```

```
3560 PROCshuffle
3570 ENDPROC
3580 :
3590 DEF PROCclear:VDU4
3600 PRINTTAB(0,19);SPC(19);
3610 FOR Y%=2 TO 17
3620 PRINTTAB(0,Y%);SPC(12)
3630 NEXT
3640 ENDPROC
3650 :
3660 DEF PROCtitle
3670 VDU5:GCOL0,3
3680 M%=0
3690 FOR Z%=900 TO 480 STEP-40
3700 M%=M%+1
3710 MOVE0,Z%:PRINT;N$(M%)
3720 NEXT
3730 FOR Z%=740 TO 500 STEP-40
3740 M%=M%+1
3750 MOVE 832,Z%:PRINT;N$(M%)
3760 NEXT
3770 ENDPROC
3780 :
3790 DEF PROCmaster
3800 VDU29,1024;840;18,0,3
3810 MOVE-100,-60:MOVE100,-60
3820 PLOT85,100,60:MOVE100,60
3830 MOVE-100,60:PLOT85,-100,-60
3840 MC%=RND(3)-1:MD%=RND(5)
3850 GCOL0,MC%:MOVE-64,32:PRINTA$(MD%)
3860 GCOL0,0:MOVE(MC%*70-100),64
3870 PRINTCHR$244
3880 ENDPROC
3890 :
3900 DEF PROCping
3910 FOR I%=101 TO 197 STEP 8
3920 SOUND1,1,I%,3:NEXT
3930 ENDPROC
3940 DEF PROCptr(M%)
3950 GCOL0,M%
3960 MOVEJ%*384-192,452:PRINT;B$
3970 ENDPROC
3980 :
3990 DEF PROCmoveptr
4000 SOUND 1,S%,64,3
4010 J%=J%+1:IF J%=4 J%=1
4020 PROCptr(3)
4030 ENDPROC
4040 :
4050 DATA Brilliant,3200,Excellent,3000
4060 DATA Good,2900,Average,2800
4070 DATA Fair,2500,Unlucky,2000,?,0
```

Wordwise User's Notebook (continued from page 28)

7) Mark current position in text (with @, or any other suitable symbol)
8) Goto start of text
9) f4 - find character
10) f3 (character to find, marker)
11) Delete marker
12) f4 - find character
13) f3 (character to find, marker)
14) Delete marker
15) Goto start of text
16) f4 - find character
17) @ (character to find)
18) Delete character

As you can see, quite a lot can be done with just one key definition! And of course this definition can be used just as well in WW as in WW+ by omitting the 'Y' (step 5) response from the definition (in WW+ the 'Y' only applies if an existing file is to be overwritten).

SINGLE BLOCK COPYING

The usual method in both WW and WW+ is to delimit the block to be copied with f3 markers, position the cursor where the copy is to go, then press f9, and hey presto, a duplicate chunk of text; the markers then have to be removed in subsequent separate operations. However, the copy and marker delete operations can all be performed at one stroke in a similar manner to Save Marked Text, using the following:

```
*KEY11"|!)@|!|O|!$|!#|A|!$|!#|A|!|O|
!$@|A"
```

in which the Copy key has been turned into a block copy key.

But that's far from being the end of the story. There's quite a lot more power to be extracted from the 'soft' keys, and although WW and WW+ do impose their restrictions on what can be achieved, the real limit is determined by your own imagination.

HINTS and tips HINTS and tips HINTS and tips HINTS and tips HINTS and tips

Please do keep sending in your hints for all BBC and Master computers. Don't forget, if your hint gets published, there's a financial reward.

FAME AND FORTUNE ON FILM

Bernard Beeston

Here is a hint that may interest some readers, though it is a little out of the ordinary for BEEBUG. My suggestion is that as the video and RF outputs from the Beeb, Electron and Master can be displayed on a standard television or monitor, the signal can also be recorded on a domestic video, making the computer a convenient source of titles and graphics for video tapes. This would be of particular interest to camcorder users. The titles can be recorded straight onto tape in the domestic recorder, and then the tape can be transferred to the camcorder (without rewinding) for recording of the subject matter. If the camcorder has input sockets, then the camcorder itself can be connected to the computer. I would recommend using the video socket from the computer rather than the RF output, as the picture quality is better and the RF output can cause problems on some video recorders unless an in-line attenuator is used. Software to produce titles on a Beeb is scarce, but a few lines of Basic can produce title screens to cheer up any home made film.

DFS/ADFS DISC PROBLEMS

Gareth Leyshon

In Mr Polak's hint for formatting a disc to both ADFS and DFS formats (see Hints and Tips, Vol.11 No.2), there are a couple of problems. First, the Aform formatting program will only accept drives 0, 1, A or B, so you can't format drive 2 of the disc using Aform. The solution is to format side 0 of the disc to ADFS (using Aform, with size S(mall) for 40 track or M(edium) for 80 track), and then format side 2 to DFS. Back up your original DFS disc to side 2 of this dual format disc, and then use the *Copyfiles* program to copy from side 2 (DFS) to side 0 (ADFS). Now do the following:

```
*FADFS
*MOUNT 0
*OPT 4 3
```

and you should be able to boot up the dual format disc as an ADFS disc. There was also a problem with setting up the *!Boot* file on the disc, though if the above method is used the problem no longer occurs. When the *!Boot* file on the DFS side is executed, then the *FADFS command prevents the rest of the file being read. The correct solution is to set up a function key to insert the characters into the keyboard buffer, as follows:

```
*KEY 0 *FADFS|M*MOUNT 0|M*EXEC !Boot|M
*FX 138,0,128
```

This enables the ADFS side to be booted.

(Technical Editor's Note: My apologies for the second of the above blunders, which was due to my own over-enthusiastic editing rather than Mr Polak's method.)

WORDWISE CODES

L.Tanner

Some interesting results can be obtained by using the following control codes in Wordwise text. In each case the usual *F1OC(code)F2* code sequence is used, but without commas (e.g. to enter code 3001, press F1, enter OC3001 and press F2). Experimenting with four figure codes starting with 27, 28, 55 etc. can produce similar results, but to work out all the combinations would need a search through the printer manual. I have tried codes in the range 2962-3093 on an NEC P2200 printer, which to all intents and purposes is Epson compatible, and some very useful characters can be obtained using this method.

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 40p per word (inclusive of VAT) and these will be featured separately. Please send all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS.

WANTED: Bruce Smith book - BBC Micro Assembler Workshop in good condition, tape to disc utility for BBC Master "Master Replay" or others, BBC magazines and discs. Tel. (0703) 823924 day (0703) 780787 eves.

BBC B with Wordwise+ twin 40/80T drive, CUB colour monitor, manuals, joystick, Brother HR15 printer £320 may split, CST Procyon IEEE Interfae £15, 9" NEC mono monitor £25, Archimedes R140 with Taxan 770+ multisync monitor £995, Wordwise ROM £12, ATPL ROM board £20, mouse as new £30, Dr Who's Mines of Terror ROM £10, Philips colour monitor £150. Tel. (0483) 480632.

BEEBUG Master ROM in original box £20, Morley Teletext adaptor, suit BBC B or BBC Master £55. Tel. (0253) 767987 after 7pm.

Juki daisywheel printer, high quality print, many fonts £55, Z80 2nd processor with original CP/M software and many extras £55, 11x1987 Acorn User mags (Aug missing) £11, 12x1986/7 Micro User mags in binder £12. Tel. (0252) 510486.

Master 128 £150, Turbo board (65c102) second processor £25, 512 board second processor upgraded to 1Mb and Gem mouse etc. with DOS+ v1.2 and v2.1 £125, Dabs Press Side Writer (5.25 40/80T) £5, StardataBase Database ROM and utilities disc £5, C.C. Print Master ROM £15, C.C. Interchart ROM £10, 4th Dimension Holed Out £5, BBC soft Vu-Type Professional £5. Discs and ROMs are all originals and with full documentation. Will sell seperately or altogether, make me an offer. Tel. 081-684 99340 eves or w/ends.

BBC bits, Z80 2nd processor £45, 512 co-processor with Gem mouse and Dr Dos 2.1 £80, Microvitec monitor £75, Prism 2000 modem £15, also lots of software. Tel. (0372) 740678.

Master 512 with mouse, Opus High Res. monitor, Cumana switchable dual disc drives with PSU, twin stand, fitted ROMs; Interword, Intersheet, Interchart, Interbase, Commsoft, Wordwise Plus four dual cartridges (two by Care Electronics), all manuals for hardware and software, The Account Book v.2 (Apricote Studios), Astaad CAD package, Speech, Gemini Office Mate, Gemini Office Master all complete with manuals, Assembley Language Programming for the BBC Micro by Ian Birnbaum, BEEBUG magazines April '84-current. Offers invited, buyer collects. Tel. (0582) 410773.

Study Electronics on the BBC Micro

An interactive approach to learning.
Four program titles available. 'Introduction to *Electronics Principles*', '*Electronics Mathematics*'
'Digital Techniques
and now 'Programming for Electronics'.
Programs include theory, examples, self test questions, formulae, charts and circuit diagrams.
User inputs and calculated outputs.

£29.95 each + £2 p&p.

Cheque or Postal Order to;
EPT Educational Software,
Pump House, Lockram Lane, Witham,
Essex CM8 2BJ
Please state BBC B/Master series and disc size.

Wish something new was happening for your BBC Micro, Master or Electron?
Something is!

Snacker - One of the latest batch of additions to the catalogue, and probably the most professional PD game yet!
Send £1.50 for catalogue and sampler disc to;

BBC PD, 18 Carlton Close, Blackrod,
Bolton, BL6 5DL

Make cheques payable to;
'A Blundell' or send an A5 s.a.e for more details
(Please state disc size and format)

Master 128, 512 co-processor (DOS+, GEM, mouse), Acorn amber monitor, dual 5.25" and 3.5" drives, 6502 second processor, Bitstick £300, Citizen 120D 9 pin dot matrix printer £50, *WS3000 modem V21/22/23, Hayes £60.* Tel. 081-698 3772.

Master 128 Turbo with upgrade OS chip, plus Overview package, ADT and Ultracalc ROMs, *Windomatic utility (disc) for Viewsheet windows editing*, MasterCopy disc (ADFS/DFS), three dual ROM cartridges, Delta 3B single joystick and Elite disc, all manuals including Welcome guide and ref. manuals 1&2, Dabs Guides to View family £200, Cumana dual 5.25" 40/80T switchable disc drives with PSU £75, Philips monochrome (white screen) monitor £35, Baksoft Dos-Copy disc (BBC/MS-DOS) £15. Tel. 071-267 1533.

BBC B software; View 3 £20, Viewsheet £17, Beebaid £10, Helping Hand £10. Tel. 081-318 5155.

All for Master Compact; P.I.A.S £3, Citadel £3, Colossus Chess 4.0 £5, Elite £5, also printer cable for Panasonic KXP1080 130cm long £3. Tel. (0482) 46812.

WANTED: A copy of the children's book "The BBC Micro" by Richard and David Graves published by Kingfisher Books Ltd. Tel. (0380) 850454.

Turbo board (65C102) for Master 128 £55, Datachat 1223 modem with ROM software £33, Micro User and BEEBUG mags 50p each (p&p extra). Tel. 051-606 0289.

BEEBUG - complete set of issues todate, very good to mint condition. Offers? Tel. (0742) 467609.

BBC Master games; Cholo, UIM, Repton Infinity, Exile £5 each, Red Arrows, Strike Force Harrier, Spitfire 40, Firetrack, Ransack, Cheat it again Joe 1/2/3, By Fair Means or Foul, Shark, Zenon, The Hunt, Bone Cruncher, Grand Prix Construction Set, Spellbinder, Superior Collection Vol.1 Codename Droid, Spycat, Repton Around the World, Star Wars £4 each, Imogen, Ravenskull, Citadel, Galaforce, Repton 2&3, Stryker's Run, Thrust, Impossible Mission £3 each all prices o.n.o. Tel. (0703) 261789 eves or w/ends.

HELP WITH VIEWSHEET

I am looking for help with ViewSheet which I use on my Master 128. I use a complex spreadsheet with six linking files to transfer data from one sheet to the next. The old sheet is then saved under a new name.

The problem arises when reloading old sheets when the 'read' slots read the current data and mess everything up. I have overcome this by deleting the 'read' and entering the value before saving, but this is very time consuming, as it also has to be done on the 'write' slots.

The DataSheet application in the May issue of BEEBUG (Vol.11 No.1) has a function whereby pressing f0 twice deletes the formulae and replaces it with the last calculated value. Is there any way of doing this in ViewSheet?

S.L.Campbell

We will pass any information on to Mr. Campbell if sent c/o BEEBUG.

IMPROVING THE ADFS DISC ORGANISER

Thank you for publishing my letter concerning the *Unlock* and *Re-Lock* modifications to the above program (see BEEBUG Vol.10 Nos.7 & 8, and Vol.10 No.10). I have since found that if you *CDIR a new directory and try to move a file into the blank directory, an error message is generated because at line 5700 there are no files for OSCLI("ACCESS") to access. The solution is to add line 5705:

```
5705 PROCpanelinfo(driveto$)
```

I also offer a couple more enhancements. As I only use Wordwise and View Professional, I have added both of these to the submenu in line 2670 so that they can be entered directly from the Organiser. The additional or modified lines are as follows:

```
2670 ......y$;"Wordwise.9"'y$;"ViewProf.0"
2680 REPEAT:key$=GET$:
UNTIL INSTR("1234567890",key$)>0
2803 IF key$="9" THEN *DIR$
2804 IF key$="9" THEN *DIR WORDWISE
2805 IF key$="9" THEN *WORDWISE
2806 IF key$="0" THEN *DIR$
2807 IF key$="0" THEN *DIR VIEWPROF
2808 IF key$="0" THEN *VP
4580 IF type$="BASIC" OR type$="Wordwise" OR
type$="Text/VuPro" THEN 4610
4680 IF type$="WordWise" THEN *DIR$
4681 IF type$="WordWise" THEN *DIR WORDWISE
4682 IF type$="WordWise" THEN *WORDWISE
4685 IF type$="Text/VuPro" THEN *DIR$
4686 IF type$="Text/VuPro" THEN *DIR VIEWPROF
4687 IF type$="Text/VuPro" THEN *VP
5410 IF char%=0 THEN ="Data" ELSE ="Text/VuPro"
```

It is important to note that if you do not create directories in advance for WordWise and ViewProf, an error will be generated.

Ian Crawford

And Bernard Hickman writes:

The following four lines, added to the Disc Organiser program, will enable the entry of 0, 1, 4, or 5 as drive numbers allowing the use of a hard disc:

```
1310 PRINT"c$;"Drive for left panel
(0/1/4/5)?"y$;"0";:VDU8:temp=GET
1320 IF temp=49 THEN PRINT"1":ldrive%=1
ELSE IF temp=52 THEN PRINT"4":ldrive%=4
ELSE IF temp=53 THEN PRINT"5":ldrive%=5
ELSE PRINT"0":ldrive%=0
1340 PRINT"c$;"Drive for right panel
(0/1/4/5)?"y$;"0";:VDU8:temp=GET
1350 IF temp=49 THEN PRINT"1":rdrive%=1
ELSE IF temp=52 THEN PRINT"4":rdrive%=4
ELSE IF temp=53 THEN PRINT"5":rdrive%=5
ELSE PRINT"0":rdrive%=0
```

Referring to the original program, Master 128 users should remove the +" I" (the initialise ROM command for the Master Compact) from the end of line 4770, or "Bad command" will be returned. Further, the use of OPENUP in lines 5340 and 5380 returns an ADFS "Access violation" error unless the "W" attribute is set. Changing the OPENUP to OPENIN will enable files with access LR and R to be examined too.

Bernard Hickman

BEEBUG MEMBERSHIP

SPECIAL OFFERS TO BEEBUG MEMBERS

BEEBUG members can purchase BEEBUG and RISC Developments' products at special discounted prices. For additional offers see the centrefold in the magazine.

Code	Product	Price
0077b	C - Stand Alone Generator	14.56
0081b	Masterfile ADFS M128 80 T	16.86
0024b	Masterfile DFS 40 T	16.86
0025b	Masterfile DFS 80 T	16.86
0074b	Beebug C 40 Track	45.21
0075b	Beebug C 80 Track	45.21
0084b	Command	29.88
0073b	Command(Hayes compatible)	29.88
0053b	Dumpmaster II	23.76
0004b	Exmon II	24.52
0087b	Master ROM	29.88
1421b	BEEBUG Binder	4.20

SUBSCRIPTION RATES

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also

BEEBUG		BEEBUG & RISC USER
£18.40	1 year UK, BFPO, Ch.I	£28.90
£27.50	Rest of Europe & Eire	£42.90
£33.50	Middle East	£53.10
£36.50	Americas & Africa	£58.40
£39.50	Elsewhere	£62.50

BACK ISSUE PRICES (per issue) 1 July 1991

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. There is no VAT on magazines.

Volume	Magazine	5"Disc	3.5"Disc
6	£1.00	£3.00	£3.00
7	£1.10	£3.50	£3.50
8	£1.30	£4.00	£4.00
9	£1.60	£4.75	£4.75
10	£1.90	£4.75	£4.75

POST AND PACKING

Please add the cost of p&p when ordering. When ordering several items use the highest price code, plus half the price of each subsequent code.

Stock Code	UK, BFPO Ch.I	Europe, Eire	Americas,Africa, Mid East	Elsewhere
a	£ 1.00	£ 1.60	£ 2.40	£ 2.60
b	£ 2.00	£ 3.00	£ 5.00	£ 5.50

BEEBUG 117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 860263
Manned Mon-Fri 9am-5pm (for orders only 9am-6pm and 9am-5pm Saturdays)
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by RISC Developments Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: Mark Moxon
Editorial Assistance: Marshal Anderson
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Subscriptions: Sue Baxter
Managing Editor: Sheridan Williams

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, RISC Developments Limited.

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

RISC Developments Ltd (c) 1992

ISSN - 0263 - 7561
Printed by Arlon Printers (0923) 268328

Magazine Disc

August/September 1992

DESIGNING NLQ CHARACTERS (PART 1) - The program Design allows NLQ printer characters to be designed for those printers that allow fonts to be downloaded.

HIGH SECURITY CIPHERS (PART 2) - The Hush program is a version of the cipher system that allows lower case entry, and the Key program enables suitable keys to be generated for the ciphers.

WORDWISE USER'S NOTEBOOK - this disc includes seven Wordwise Plus segment programs which make use of the function keys. Example uses include implementing a highlighting and marking system within Wordwise Plus, as well as quick file saving and a menu system.

EMBLEMS GAME - The Emblems game is in the style of the dice game Yahtzee, except Emblems uses cards with five different emblems in three different colours.

MR TOAD'S MACRO ROM - This sideways ROM image provides a macro facility whereby text strings can be stored and entered into word processors (for example) via a simple key sequence.

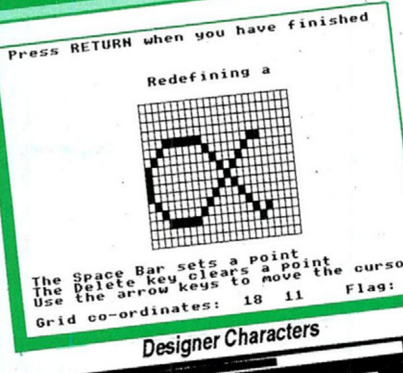
DYNAMIC MEMORY WINDOW - This utility from Vol.4 No.5 displays a dynamic memory dump, i.e. one that is constantly updated. The display can be active while other tasks (such as program editing) are being performed.

BEEBUG WORKSHOP: LINKED LISTS (PART 2) - The program on this disc (which is repeated from last month) implements the data structure of linked lists. Using this program you can enter data, delete data, search for data and display data, all using linked lists.

FLASHING COLOURS IN MODE 1 - This program allows flashing colours to be displayed in all modes except mode 7. The text doesn't just flash, but it cycles through coloured sequences by using interrupts.

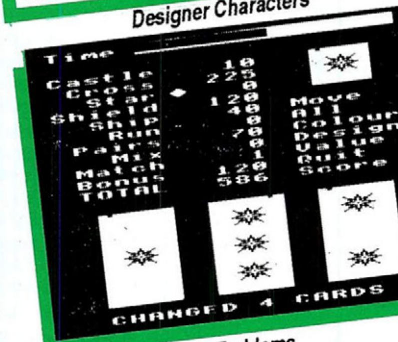
CHESS DATABASE - This program, a bonus item on the disc, allows details of Chess games to be edited and stored to disc, and stored games can be stepped through using a graphical Chess display.

MAGSCAN DATA - Bibliography for this issue (Vol.11 No.4).



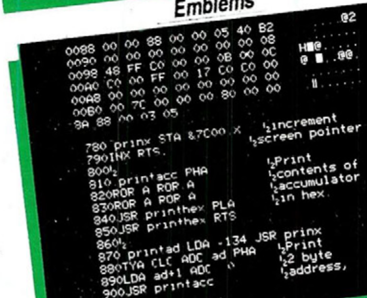
```
Press RETURN when you have finished

Redefining a

The Space Bar sets a point
The Delete key clears a point
Use the arrow keys to move the curso

Grid co-ordinates:   18  11       Flag:
```

Designer Characters



```
Time
Castle     10
Cross     225
Star        0
Shield    120      Move
Ship       40      All
Run         0      Colour
Pairs      70      Design
Mix         0      Value
Match       1      Quit
Bonus     120      Score
TOTAL     586

CHANGED 4 CARDS
```

Emblems



```
                                          @2
0088 00 00 88 00 00 05 40 B2
0090 00 00 00 00 00 00 00 08       H█@....
0098 48 FF C0 00 00 0B 00 0C       @ █..@@.
00A0 C0 00 FF 00 17 C0 C0 00
00A8 00 00 00 00 00 00 00 00         ‖.....
00B0 00 7C 00 00 00 80 00 00
8A 88 00 03 05
                                ½increment
 780 prinx STA &7C00 X          ½screen pointer
 790INX RTS.
 800½                           ½Print
 810 printacc PHA               ½contents of
 820ROR A ROR A                 ½accumulator
 830ROR A ROR A                 ½in hex.
 840JSR printhex PLA
 850JSR printhex RTS
 860½
 870 printad LDA -134 JSR prinx ½Print
 880TYA CLC ADC ad PHA          ½2 byte
 890LDA ad+1 ADC                ½address,
 900JSR printacc

Escape at line 900
```

Dynamic Memory

ALL THIS FOR £4.75 (5.25" & 3.5" DISC) + £1 P&P (50P FOR EACH ADDITIONAL ITEM)

Back issues (5.25" and 3.5" discs) from Vol.6 No.1 available - see Membership Page.

DISC (5.25" or 3.5") SUBSCRIPTION RATES	UK ONLY	OVERSEAS
6 months (5 issues)	£25.50	£30.00
12 months (10 issues)	£50.00	£56.00

Prices are inclusive of VAT and postage as applicable Sterling only please

RISC Developments, 117 Hatfield Road, St.Albans, Herts AL1 4JS

Upgrading to an Archimedes

We know that many BEEBUG readers have already upgraded to an Archimedes, and no doubt many more will choose to follow a similar route. For their benefit we offer our advice to help them make a sensible decision on whether to upgrade and if so, what path to take.

Any prices quoted relate to our associated company Beebug Ltd., but note that all prices, particularly those on trade-ins and secondhand items, are likely to change without notice. You should always telephone or write for the latest information.

Archimedes A5000

What System to Choose

All Archimedes systems still use the RISC OS 2 operating system, except the A5000 and A4 which use RISC OS 3. Based on the experience of existing users, we would strongly recommend a minimum of 2Mb of RAM. Most users find a hard disc adds significantly to the convenience of using an Archimedes, but you can always add a low-cost hard drive later, and more memory, but check on the likely price of future expansions - it is not necessarily the same for all machines. If you might be interested in more specialised add-ons (scanners, digitisers,etc.) then check the expansion capability of your preferred system.

Compatibility and Transferability

You will need to decide to what extent you wish to continue to use existing discs and disc drives on an Archimedes. An Archimedes and a BBC micro can be directly connected for transfer of files. You can also connect a 5.25" drive to an Archimedes via an additional interface to continue to access 5.25" discs (ADFS format). Our DFS Reader will also allow files to be transferred to the Arc from DFS format discs. Much software will run directly on a Archimedes, or via the 6502 emulator. However, consider this carefully; in our experience, despite prior misgivings, most Archimedes users find that they rapidly adjust to the Desktop environment of the Archimedes, and quickly abandon the software and data of their old system after an initial period.

Software for the Archimedes

The Archimedes is supplied complete with a range of basic applications software. Before embarking on any further purchases it may be better to familiarise yourself with the new machine. Most users look for a word processor (or DTP package), maybe a spreadsheet, or a database, plus other more specialist software. We cannot give detailed guidance here, but back issues of RISC User contain a wealth of useful information - we can advise on suitable issues.

General Advice

It is advisable to discuss your requirements with the BEEBUG technical team before making a final decision on what you want. Try to anticipate future expansion needs at the outset. Note that Archimedes systems (except the A5000) are priced without monitor. The details given in the table are minimum specifications of the different Archimedes models. Note that unlike a BBC micro, there is seldom much need for a second floppy disc drive on an Archimedes.

A3000 Learning Curve

It may also be possible to trade in an existing monitor and/or disc drive, but check if your existing monitor is suitable for use with an Archimedes first. You may find it better to advertise your BBC system in BEEBUG and sell privately - this applies particularly to any software and hardware add-ons which cannot be accepted for a trade-in. In future, all personal ads for Archimedes systems in RISC User will also be included in BEEBUG. You may also defer a trade-in until a later date provided you make this clear at the time of purchase.

Archimedes Systems - Typical or Current Prices

		Secondhand	New
+	A310 1Mb RAM	£350	
+	A410 1Mb RAM	£590	
+	A420 2Mb RAM, 20Mb hard drive	£675	
+	A440 4Mb RAM, 40Mb hard drive	£750	
+ *	A3000 1Mb RAM	£425	£ 703.83
*	A5000, 2Mb RAM, 40Mb hard drive		£1761.33
+ *	Acorn standard colour monitor	£145	£ 234.94

All systems above include a single floppy disc drive.
New (*) and secondhand (+) - all prices inc. VAT.
The A5000 price includes a multiscan colour monitor.

BBC Micros - Typical Trade-in Prices

Model B (Issue 7)	£ 35
Model B (issue 7) + DFS	£ 75
Master 128	£125
Master Compact	£ 50

For further information on all Archimedes systems contact:
BEEBUG Ltd. 117 Hatfield Road, St Albans, Herts AL1 4JS. Tel. 0727 40303 Fax 0727 860263